Fehlerlokalisierung in prozessorinternen Kommunikationsnetzen für Vielkern-Prozessoren

Dissertation
zur Erlangung des akademischen Grades eines
Doktor rer. nat.
der Fakultät für Angewandte Informatik der Universität Augsburg

Arne Garbade

24.02.2014

eingereicht von
Dipl.-Inf. Arne Garbade

Erstgutachter:	Prof. Dr. rer. nat. Theo Ungerer
Zweitgutachter:	Prof. Dr.-Ing. Rudi Knorr
Tag der mündlichen Prüfung:	16. Mai 2014

Abstract

Future many-core processors pose a challenging demand to both, hardware architects and software architects. The ever increasing integration density of transistors and interconnects give rise to on-chip error-rates. This trend is amplified by manufacturing process variations or due to diverse aging phenomena. Simultaneously, strict energy budgets force task schedulers to take advantage of data locality in order to minimize on-chip communication and thereby energy dissipation. However, a smart task placement is only possible with prober information regarding hardware errors occurring during runtime. Regardless the fact, that a broad research has been done for both domains on-chip fault tolerance techniques and task scheduling algorithms, there is missing a mechanism that interfaces with both domains to connect them. This thesis picks up on this missing interface and presents a lightweight and decentralized method to localize faults within the on-chip interconnection network and to propagate the gathered state information to the scheduling unit. This localization method bases on the TERAFLUX-Microprocessor architecture and is part of the fault tolerance unit within this architecture. The heart of the localization technique is the timing behavior of the status messages which are transmitted from the monitored processor cores to the fault tolerance unit. It will be shown that single faults within the network can be localized precisely. For multiple faults, the approach gains localization rates up to 98% of coverage. Additionally, this work presents a new routing strategy which relaxes the impact of status messages on application message traversal times. The routing strategy broadens the workload of status messages over the network and hence lowering the maximum delays by 30%.

Kurzfassung

Zukünftige Vielkernprozessoren stellen große Herausforderungen an die zugrunde liegenden Hardware-Architekturen. Die steigende Integrationsdichte von Transistoren und Leiterbahnen, erhöht das Risiko von Fehlern auf dem Prozessor-Chip. Gleichzeitig zwingen strikte Energie-Budgets der Prozessoren, zu mehr Umsicht bei der Platzierung von Prozessen auf den Kernen, um Datenlokalität der Software möglichst gut auszunutzen und so Kommunikation einzusparen. Eine intelligente Platzierung ist aber nur dann möglich, wenn Fehler, die zur Laufzeit des Prozessors auftreten, der Prozessverwaltung bekannt sind. Obwohl bereits ein breites Spektrum an Fehlertoleranztechniken und Prozessverwaltungen entwickelt wurden, fehlt ein Mechanismus, der beide Bereiche verbindet. Diese Dissertationsarbeit greift die Lücke auf und präsentiert ein leichtgewichtiges, dezentrales Verfahren, um fehlerhafte Komponenten im Kommunikationsnetz zu lokalisieren und den aktuellen Zustand des Netzes an die Verwaltungseinheiten zu propagieren. Das Lokalisierungsverfahren basiert auf der TERAFLUX-Prozessorarchitektur und ist Teil der Fehlertoleranzeinheit dieses Prozessorentwurfs. Kernstück der Lokalisierung ist das zeitliche Verhalten der Statusnachrichten, die von den überwachten Prozessorkernen an die Fehlertoleranzeinheit übertragen werden. Diese Arbeit zeigt, dass einzelne Fehler präzise lokalisiert werden können. Bei multiplen Fehlern liegt die Lokalisierungsrate bei bis zu 98%. Gleichzeitig stellt diese Arbeit eine neue Routing-Strategie vor, um den Einfluss auf Anwendungsnachrichten durch Statusnachrichten zu minimieren. Mit der verbesserten Lastverteilung der Statusnachrichten konnten die maximalen Wartezeiten für Anwendungsnachrichten um bis zu 30% reduziert werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACK	Acknowledgement: Positive Bestätiung eines Nachrichtenempfangs.
$\mathcal{F}$	Faltungsoperator, um M^V nach M^F zu transformieren.
FDU	Fault Detection Unit: Dezentrale Fehlertoleranzeinheit in der TERAFLUX-Architektur.
M	FDU interne Statusmatrizen: Speichert die *verdächtigten*, bzw. die *ausgefallen* Komponenten des Kommunikationsnetzes.
MPR	Minimal-Path Routing: Ein Prinzip, bei dem stets der kürzeste Pfad zwischen Sender und Empfänger gewählt wird.
NACK	Negative Acknowledgement: Negative Bestätiung eines Nachrichtenempfangs.
QoS	Quality of Service: Prioritätenbasierende Regulierung der Zugriffs auf geteilte Ressourcen.
TDMA	Time Devision Multiple Access: Regulierung des Zugriffs auf geteilte Ressourcen.
TSU	Thread Scheduling Unit: Dezentrale Prozessverwaltung in der TERAFLUX-Architektur.
VB	Verbindungsbreite: Die Bandbreite einer Verbindungsleitung pro Netztakt.
WAL	Wrap-Around Lines: Spezielle Verbindungsleitungen toroidaler Netztopologien.

1. Einleitung

Seit fast einem halben Jahrhundert gilt das Mooresche Gesetz, nach dem sich (je nach heutiger Deutung) die Leistungsfähigkeit eines Computerchips etwa alle 18 bis 24 Monate verdoppelt [Moore 1965]. Seitdem entwickeln Industrie und akademische Forschung Lösungen, um diesen Trend aufrechtzuhalten. Zu den jüngsten Änderungen im Architekturbereich gehört der Paradigmenwechsel vom *Einzelkernprozessor* über die *Mehrkernprozessoren* bis hin zu den *Vielkernprozessoren* [Borkar 2007]. Es wird erwartet, dass innerhalb der nächsten fünf bis zehn Jahre ein Prozessor in der Lage ist, mehr als 1000 Allzweckkerne (General Purpose CPUs) zu beherbergen. Motiviert wird diese Annahme durch zwei Faktoren. Zum einen ermöglicht die fortwährende Miniaturisierung der Transistoren immer mehr Logik und Speicher auf einem Chip zu platzieren. Zum anderen ist das Verhältnis zwischen Leistungsfähigkeit und Energieverbrauch bei tausenden einfachen Prozessorkernen deutlich besser, als bei wenigen hochkomplexen [Borkar und Chien 2011]. Mit der Miniaturisierung der Transistoren und dem hohen Grad der Parallelität rücken jedoch neue Probleme ins Licht. Zum einen wird erwartet, dass es für die Prozessorhersteller immer schwieriger wird, die Transistoren langfristig "haltbar" zu machen. Zum anderen stellt sich die Frage, wie man die massive Parallelität des Prozessors effizient ausnutzen kann.

Die Haltbarkeit der Transistoren wird vor allem auf deren Größe zurückgeführt. Bei aktuellen Fertigungstechnologien, schrumpft der Transistor soweit, dass er nur noch aus wenigen Hundert Atomreihen besteht [Constantinescu 2003]. Selbst geringe werkzeugbedingte Fertigungsschwankungen bei der Herstellung sorgen für Ungenauigkeiten, die bei diesen kleinen Transistorgrößen zu verhältnismäßig starken Variationen führen. Zwei benachbarte Transistoren können durch diese Variationen bereits bei der Herstellung des Chips unterschiedliche elektrische Eigenschaften zeigen [Borkar 2005]. Die Folge sind Unterschiede bei den Schaltzeitpunkten und unterschiedlich starke Leckströme. Zusätzlich tritt ein weiterer Effekt während der Laufzeit des Prozessors auf. Durch die Elektronenwanderung bei fließendem Strom wird auch metallisches Material im Leiter transportiert - auch als *Elektromigration* bekannt. Dort, wo der Leiter abgetragen wird, kann es zunächst zu erhöhtem Widerstand in einer Signalleitung kommen, gefolgt von einem vollständigen Bruch an der Leiterbahn. An den Stellen, an denen sich das Material ablagert, entstehen *Buckel* im Leiter. Diese können Isolationsschichten am Rand der Leiterbahn verdrängen, was anfangs zu verstärktem Nebensprechen auf den Signalleitungen führt. Lagert sich mehr leitfähiges Material an dieser Stelle ab, droht sogar ein Kurzschluss.

Diese und weitere Effekte werden in zukünftigen Prozessorgenerationen die Zuverlässigkeit der Chips stark beeinflussen [Borkar 2005], sodass die erwartete Anzahl der Fehler im Prozessor deutlich zunehmen wird.

Die zweite Herausforderung bei den zukünftigen Prozessoren ist das effiziente Ausnutzen der Parallelität unter Berücksichtigung des Energie-Budgets. Laut [Borkar und Chien 2011] werden zukünftige Prozessoren unter anderem nur dann mit einem gegebenen Energie-Budget effizient arbeiten können, wenn es gelingt die Datenlokalität kommunizierender Programmteile zu verbessern. Die Arbeiten von [Schlingmann 2011; Dziurzanski und Maka 2013] widmen sich dieser Herausforderung, indem die Programmteile intelligent auf den Chip platziert werden und somit kürzere Kommunikationswege zwischen den Programmteilen entstehen. Jedoch basieren die dazu verwendeten Algorithmen auf Zustandsinformationen des Chips. Eine wesentliche Kennzahl dieser Informationen ist der Zustand des Netzes. Wie und wann diese Informationen erzeugt werden, ist jedoch bisher nur wenig untersucht worden, und die bislang vorgeschlagenen Lösungen zeigen Schwächen.

Das effiziente Ausnutzen der Parallelität scheint auf den ersten Blick wenig mit entstehenden Fehlern auf dem Prozessor zu tun zu haben. Jedoch gibt es eine Ebene an denen beide Aspekte aufeinander treffen. Die zu erwartende dynamische Änderung des "Gesundheitszustands" des Prozessors während der Laufzeit hat direkten Einfluss auf die Zustandsinformationen, welche zur intelligenten Prozessverwaltung auf dem Prozessor benötigt werden.

Die in dieser Arbeit erworbenen Erkenntnisse basieren auf dem EU Forschungsprojekt TERAFLUX[1]. Innerhalb dieses Projektes wurden verschiedene Aspekte eines zukünftigen Prozessors mit mehr als 1000 Prozessorkernen untersucht. Neu an diesem Prozessor ist, dass die Prozessorkerne nicht wie bei Grafikprozessoren aus tausend Spezialkernen bestehen, sondern aus x86-fähigen Allzweckprozessorkernen. Das gesamte Projekt umfasst dabei die folgenden hierarchisch aufgespannten Forschungsgebiete:

- Parallele Software, Programmiermodell und Benchmarks (Oberste Ebene)
- Compiler
- Datenflussbasierende Ausführungsmodelle, Speichermodelle und Scheduling
- Feingranulare Prozessverwaltung und Fehlertoleranz
- Hardware und Simulation (Unterste Ebene)

Diese Dissertationsarbeit gliedert sich in den Bereich der Fehlertoleranz ein und stellt im Speziellen ein Fehlertoleranzverfahren vor, das Informationen aus der Hardware-Ebene aufnimmt, auswertet und an höhere Ebenen wie das Betriebssystem und dessen Scheduler weitergibt. Das Ziel ist es den Spalt zwischen Fehlererkennung und intelligenter Prozessverwaltung zu schließen und gleichzeitig die gegebene Architektur des TERAFLUX-

[1] http://www.teraflux.eu

Prozessors möglichst unverändert zu belassen. Hierfür wurden zwei Meilensteine definiert:

(i) Entwurf, Implementierung und Evaluierung einer verbesserten Lastverteilung im Kommunikationsnetz in Bezug auf Statusnachrichten (Heartbeat-Nachrichten) von Prozessorkernen.

(ii) Extraktion von Fehlerinformationen aus dem zeitlichen Verhalten der Statusnachrichten, zur Erstellung eines Zustandsbildes des Kommunikationsnetzes.

Die Dissertationsarbeit bildet damit einen wichtigen Baustein für Betriebssyteme zur intelligenten Prozessverwaltung/Verteilung [Schlingmann 2011; Dziurzanski und Maka 2013] auf künftigen Vielkernprozessoren.

Die restliche Dissertationsarbeit gliedert sich wie folgt. Kapitel 2 beinhaltet die Grundlagen dieser Arbeit. Es geht dabei allgemein auf prozessorinterne Kommunikationsnetze ein, beschreibt die aktuellen Arbeiten zu Fehlertoleranztechniken solcher Netze und skizziert die TERAFLUX-Architektur aus Sicht der Fehlertoleranz. Im dritten Kapitel wird der erste Meilenstein diskutiert, wobei zunächst auf die Anforderungen der FDU auf das System und deren Implementierung eingegangen wird. Zusätzlich werden durch Heartbeat-Nachrichten induzierte Engpässe diskutiert und wie diese anhand der neuen Routing-Strategie vermindert werden. Eine Evaluierung dieser Routing-Strategie schließt das Kapitel ab. Das Verfahren der Fehlerlokalisierung im Netz ist Bestandteil des vierten Kapitels. Es wird zunächst die Funktionsweise des Lokalisierungsverfahrens beschrieben, Anpassungen am Heartbeat-Mechanismus definiert und die zusätzlichen Kosten durch die Anpassungen diskutiert. Ferner beinhaltet das Kapitel die Untersuchung über verschiedene multiple Fehler im Netz hinsichtlich des Lokalisierungsverfahrens. Abschließend wird die 2D-Gitter Topologie durch toroidale ersetzt und dessen Auswirkungen auf das Verfahren diskutiert. Das Kapitel 5 fasst die gesamte Arbeit zusammen und gibt einen Ausblick auf weitere Forschungsideen.

2. Grundlagen

Dieses Kapitel beinhaltet die Grundlagen der Dissertationsarbeit. Zuerst werden die relevanten Grundkonzepte der prozessorinternen Kommunikation vorgestellt. Der zweite Teil richtet den Fokus auf bestehende Fehlertoleranz-Verfahren in prozessorinternen Kommunikationsnetzen. Abschließend wird die Basis-Architektur des TERAFLUX-Prozessors vorgestellt, auf der diese Dissertationsarbeit aufsetzt.

2.1. Prozessorinternes Kommunikationsnetz

Die Beschreibung eines prozessorinternen Kommunikationsnetzes wird im Folgenden in Top-Down Form vorgenommen. Zuerst werden die Netze anhand ihrer Topologien und Router-Entwürfe beschrieben. Nachdem die Hardware des Netzes erläutert wurde, folgt die Beschreibung der verwendeten Verfahren zur Flusskontrolle und der Routing-Strategien.

2.1.1. Netztopologie

Die vier häufigsten Netztopologien für die prozessorinterne Kommunikation ist der Bus, die Baumstruktur, die Sternstruktur und das 2D-Gitter (siehe Abbildung 2.1). Für welche Topologie man sich entscheidet hängt maßgeblich von der gegebenen Zielarchitektur und den Fehlertoleranz-Aspekten ab. Deshalb werden diese vier Topologien kurz unter den Aspekten der Leistungsfähigkeit und Fehlertoleranz erörtert und die Verwendung eines 2D-Gitters als Topologie motiviert.

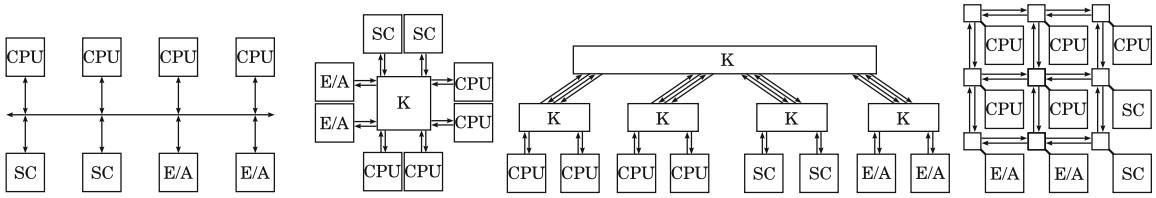
Auf einem Bus basierende Topologien waren lange die häufigste Methode um mehrere Prozessorkerne auf einem Chip miteinander zu verbinden. Solange die Anzahl der Kommunikationsteilnehmer gering ist, liefert ein Bus gute Leistung in Bezug auf Datentransfer und Energieverbrauch. Doch reicht die Leistungsfähigkeit des Busses nicht, um effizient mehr als acht Kommunikationspartnern als Kommunikationsmedium zu dienen. Abbildung 2.1(a) zeigt die typische Struktur eines auf einem Bus basierenden Netzes. Für den Aspekt der Fehlertoleranz ist die zentrale Verbindungsleitung ein *single point of failure*, bei dem bereits der erste Fehler in der Lage ist, alle Kommunikationsteilnehmer vom Kommunikationsnetz zu trennen. Aus diesen Gründen erfüllen auf einem Bus basierende Topologien weder die Leistungsansprüche eines 1000 Kern Prozessors,

noch bieten sie aus Sicht der Fehlertoleranz ausreichend Redundanz um Ausfälle von Verbindungsleitungen zu kompensieren.

Die Sterntopologie, wie in Abbildung 2.1(b) mit einer zentralen Kreuzschiene [Kim 2004] dargestellt, hat ähnliche Eigenschaften in Bezug auf Fehlertoleranz wie auf einem Bus basierende Topologien. Hier ist die Kreuzschiene an sich der *single point of failure* und beinhaltet damit die Gefahr, alle Teilnehmer mit dem ersten Fehler vom Netz zu trennen. Aus Sicht der Leistungsfähigkeit hingegen ist die Sterntopologie deutlich effizienter als ein Bus. Die Kreuzschiene ermöglicht Punkt-zu-Punkt Verbindungen, die mehrere Kommunikationspartner simultan miteinander verbinden. Die Anzahl der Teilnehmer ist aber auch hier begrenzt. Der limitierende Faktor der Kreuzschiene ist die maximale Betriebsfrequenz und ihr Platzverbrauch. Je mehr Teilnehmer an die Kreuzschiene angeschlossen werden, desto größer werden der Platzverbrauch und damit auch die Distanz, die die Signale innerhalb der Kreuzschiene zurücklegen müssen. Letzteres wirkt sich vor allem auf die maximale Betriebsfrequenz aus. Die Kommunikation durch die Kreuzschiene muss aus technischen Gründen[1] für alle Teilnehmer gleich lang dauern. Die kann bei langen Wegstrecken innerhalb der Kreuzschiene aber nur dann garantiert werden, wenn die Betriebsfrequenz gesenkt wird. Erst dann ist die Taktdauer lang genug, um alle Signale in einem Takt übertragen zu können.

Im Vergleich zu den Bus- und den Stern-Topologien liefert die Baumstruktur (Abbildung 2.1(c)) die beste Leistung in Bezug auf Betriebsfrequenz und Übertragungslatenz. Die begrenzte Anzahl der Ein- und Ausgänge der einzelnen Kreuzschienen ermöglicht hohe Betriebsfrequenzen. Mit Hilfe der wenigen Zwischenstationen durch das Netz werden sehr geringe Latenzzeiten erreicht [Adriahantenaina 2003; Flich und Bertozzi 2010]. Die maximale Anzahl der Kerne ist variabel und hängt von der Bandbreite der Verbindungsleitungen zwischen den Kreuzschienen ab. Allerdings hat auch diese Topologie schlechte Redundanzkapazitäten. Obwohl die Kommunikationswege voneinander getrennt sind, stellt die zentrale Kreuzschiene auf der obersten Stufe einen *single point of failure* dar.

[1] Der Entwurf von asynchronen Kommunikationssystemen auf Prozessoren stellte sich bisher immer als sehr schwierig heraus, da der Übergang von einer Region der Taktung A in eine Region mit Taktung B häufig zu Timing-Fehlern führt [Chen 2012].



(a) Bus-Topologie (b) Stern-Topologie (c) Baum-Topologie (d) 2D-Gitter

Abb. 2.1.: Die vier häufigsten Netztopologien auf Prozessoren: Bus, Stern, Baum und 2D-Gitter.

Darüber hinaus verwenden typische Baum-Topologien fest verdrahtete Kommunikationswege [Flich und Bertozzi 2010], die es schwer machen einen Nutzen aus den redundanten Verbindungen zu beziehen.

Das 2D-Gitter ist eine gute Wahl aus Sicht der Fehlertoleranz. Um auf fehlerhafte Verbindungsleitungen und sogar ausgefallene Router zu reagieren, bietet die redundante Struktur (Abbildung 2.1(d)) des 2D-Gitters auf natürliche Weise viele alternative Pfade durch das Kommunikationsnetz. Nach Definition von [Dally und Towles 2003] ist eine solche Topologie ein stark verbundenes Netz (strongly connected), was sowohl eine Kennzahl für den Grad der Fehlertoleranz ist als auch für die mögliche maximale Leistungsfähigkeit. Ein 2D-Gitter bietet gute Bandbreiten bezüglich der Leistungsfähigkeit, da auch die Anzahl der Verbindungsleitungen mit der Zahl der Kommunikationsteilnehmer natürlich mitwächst [Duato 2002]. Zusätzlich lassen sich mit einer entsprechenden Routing-Strategie selbst überlastete Bereiche des Netzes durch alternative Pfade umgehen. Aufgrund einer höheren durchschnittlichen Anzahl der Zwischenstationen durch das Netz ist die Leistungsfähigkeit in Bezug auf die Latenzzeit jedoch nicht so hoch wie die einer Baumstruktur. Eine aktuelle Implementierungen der 2D-Gitter Topologie ist das iMesh der TILE-Gx Prozessor Familie [Wentzlaff 2007]. Weiterführende Gitter-Topologien, wie der Torus mit noch mehr Redundanz, werden in einem eigenen Abschnitt der Fehlerlokalisierung im Abschnitt 4.7.2 diskutiert.

Neben der vorangegangenen strikten Trennung der Topologien existieren in heutigen Prozessoren auch Mischformen unterschiedlicher Topologien. Die Kernidee beim Kombinieren der Topologien liegt darin, die jeweiligen Nachteile einer bestimmten Topologie mit der einer anderen zu kompensieren. Als praktisch haben sich dabei Mischformen herausgestellt, welche die Prozessoren in eine Gruppe zusammenfassen und Gruppenintern mit einer bestimmten Topologie zu vernetzen. Die Gruppen untereinander werden unter Verwendung einer anderen Topologie miteinander vernetzt. Diese Vernetzung wird auch *Clustered-Architecture* genannt. Prominente Beispiele dieser Architektur sind Intel's *Single-Chip Cloud Computer* (SCC) [Wijngaart 2011] und Kalray's *MPPA-256* [Dinechin 2013a; Dinechin 2013b]. Der SCC gruppiert zwei Prozessorkerne zusammen und verbindet diese mit einem gemeinsamen Nachrichtenpuffer. Der Nachrichtenpuffer ist zusätzlich mit einem Router verbunden, der wiederum in einer 2D-Topologie mitx allen anderen Routern verbunden ist. Kommuniziert wird auf dem SCC gruppenintern nur über den Nachrichtenpuffer. Über Gruppengrenzen hinweg wird das Netz der Router zur Kommunikation verwendet. Der MPPA-256 verfolgt eine ähnliche Strategie. Hier werden jedoch 17 Kerne in eine Gruppe zusammengefasst. Die Gruppen sind untereinander ebenfalls mit Routern verbunden[2]. Die Verbindungen zwischen den Routern des MPPA-256 erzeugen eine toroidale Topologie und ordnen sich in die zuvor beschriebenen

[2]Über die Verbindungsart innerhalb der Gruppe ist bislang öffentlich nichts bekannt. Da aber aus verschiedenen Datenblättern [Dinechin 2013a; Dinechin 2013b] ein großer gemeinsamer Speicher im Zentrum der Gruppe liegt, liegt die Vermutung nahe, dass die Kommunikation ähnlich wie im SCC abgewickelt wird.

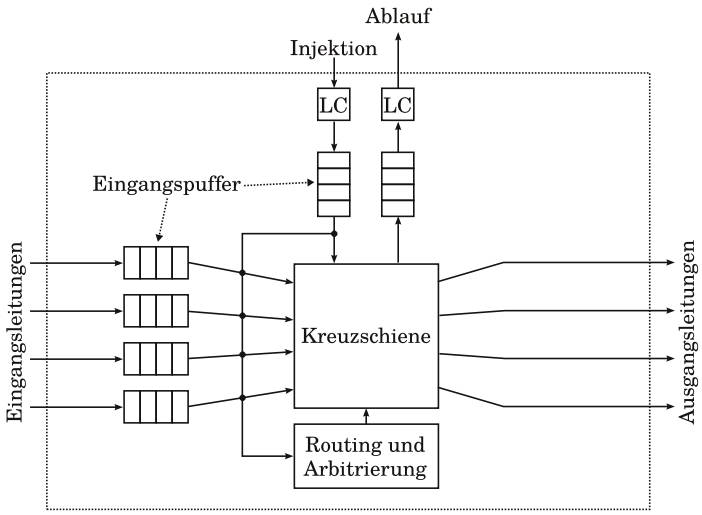


Abb. 2.2.: Generisches Router-Model [Duato 2002]

2D-Gitter Topologien ein. Zusätzlich enthält der Prozessor eine spezielle Schnittstelle, die zusätzliche MPPAs direkt miteinander verbinden kann[3].

2.1.2. Router

Der Router eines prozessorinternen Netzes stellt für die Nachrichten basierende Kommunikation zwischen zwei Kommunikationspartnern die Infrastruktur bereit. Die Kommunikationspartner bestehen in der Regel aus einem Prozessorkern, der die Kommunikation anregt, und einem Speicher-Controller, Ein- und Ausgabe-Controller oder einem anderen Prozessorkern. Die Router selbst beginnen von sich aus keine Kommunikation. Abhängig von der Netztopologie sind die Router jeweils mit einem Prozessor oder Controller und weiteren Routern verbunden. In Abbildung 2.2 ist ein generisches Router-Modell nach [Duato 2002] dargestellt. Dieses vereinfachte Modell besteht aus vier Komponenten:

(i) Der Pufferspeicher ist mit den physikalischen Eingangsleitungen verbunden, nimmt Nachrichten auf und speichert diese. Weitere Pufferspeicher an den Ausgangsleitungen sind optional und können eingesetzt werden, um beispielsweise Nachrichtenstaus besser auflösen zu können.

(ii) Die Kreuzschiene verbindet die Pufferspeicher der Eingangsleitungen mit den Ausgangsleitungen. Für maximale Leistung des Routers werden Kreuzschienen verwendet, die in der Lage sind alle vier Eingangsleitung mit den Ausgangsleitungen zu verbinden, sofern dabei keine Konflikte auftreten.

[3] Auch hier gibt es keine öffentlich zugänglichen Informationen. Jedoch verwenden Intel mit dem Quick Path Interface und AMD mit dem Hypertransport vergleichbare Ansätze, um Prozessoren miteinander zu verbinden.

(iii) Die Routing-Einheit implementiert das Protokoll, mit dem entschieden wird, auf welchem Pfad eine Nachricht durch das Netz übertragen wird. Dafür ermittelt jede Routing-Einheit lokal auf dem Router, welcher Eingangspuffer mit welcher Ausgangsleitung verbunden werden soll. Des Weiteren steuert die Routing-Einheit zusammen mit der Arbiter-Einheit die Kreuzschiene des Routers an. Die entstandene Verbindung zwischen Eingangspuffer und Ausgangsleitung wird als Kanal bezeichnet, und ist gerade bei den gepufferten Flusskontrollverfahren relevant.

(iv) Die Arbiter-Einheit löst Konflikte auf, die bei konkurrierenden Zugriffen von mehreren Nachrichten verschiedener Eingangspuffer auf eine Ausgangsleitung entstehen. Dazu implementiert die Arbiter-Einheit ein Flusskontrollverfahren, das den Zugriff auf die Ausgangsleitungen regelt. Je nach Implementierung wird dabei eine blockierte Nachricht im Pufferspeicher gehalten bis der Konflikt aufgelöst werden konnte, oder aber die Nachricht wird verworfen beziehungsweise über eine andere Ausgangsleitung weitergeleitet.

Die zusätzlich eingezeichneten Komponenten *Injektion* bzw. *Ablauf* stellen in diesem generischen Modell die Verbindungen des Routers mit einem Prozessorkern dar. Nachrichten können im gesamten Netz nur über diese Injektionsleitungen das Netz betreten. Die Nachrichten können das Netz entweder über den Ablauf verlassen oder werden vom Router verworfen, wenn sie blockiert werden.

2.1.3. Flusskontrolle

Die Flusskontrolle ist Teil der Arbiter-Einheit. Sie steuert den Zugriff der Eingangspuffer auf Ausgangsleitungen[4] und regelt wieviel Bandbreite einem Kanal zur Verfügung steht. Kommt es zu einem konkurrierenden Zugriff auf eine Ausgangsleitung, löst die Flusskontrolle den Konflikt, indem einer der Eingangspuffer verbunden wird und die restlichen konkurrierenden Eingänge blockiert werden. Potenzielles Verhungern eines Eingangspuffers wird häufig mit Hilfe von *Round Robin*-Zugriffsverfahren vermieden. Alternativ kann die Flusskontrolle die blockierten Nachrichten verwerfen oder auf einer anderen Route durch das Netz übertragen lassen. Voraussetzung für alternative Routen ist jedoch eine Routing-Strategie, die dies erlaubt, ohne dabei die Gefahr eines Deadlocks oder Livelocks zu erzeugen.

Nachrichten zu verwerfen ist in den meisten Fällen bei prozessorinternen Netzen höchst problematisch, denn eine verworfene Nachricht muss erneut vom Absender abgeschickt werden. Dies setzt aber voraus, dass der Sender alle abgesetzten Nachrichten zwischenspeichert, bis diese

(i) tatsächlich ihr Ziel erreicht hat und

(ii) der Empfänger den Empfang der Nachricht quittiert.

[4]Die Kanalbildung erfolgt mit Hilfe der Kreuzschiene.

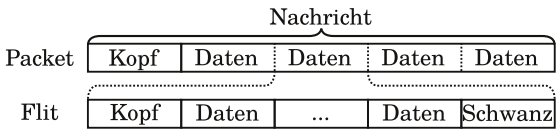


Abb. 2.3.: Zerlegung von Nachrichten in Pakete und Flits.

Das Zwischenspeichern der Nachrichten erfordert gerade bei intensiver Kommunikation viel Speicherplatz seitens des Absenders. Kleinere Puffer in den Routern, die die blockierten Nachrichten zwischenspeichern, sind daher effizienter. Ferner steigt der Datenverkehr durch Bestätigungs- oder Verlustnachrichten im Netz schnell an. Damit können die Antwortzeiten durch eine erneute Transmission schnell ansteigen.

Wie und wo blockierte Nachrichten zwischengespeichert werden, bestimmt maßgeblich die Granularität in der die Nachrichten übertragen werden. Alle Nachrichten werden grundsätzlich in Paketform durch das Netz übertragen [Dally und Towles 2001]. Ein Paket besteht dazu aus einem Paketkopf und dem eigentlichen Datensatz (Abbildung 2.3, oberer Teil). Nachrichten können auf diese Weise im Ganzen verschickt werden, wenn Flusskontrollverfahren wie das *Store-and-Forward* oder *Cut-Through* verwendet wird. Der Vorteil dieser Art des Nachrichtenversands liegt darin, dass die Kosten durch den Paketkopf (Protocol Overhead) mit wachsender Größe des Datensatzes besser amortisiert werden. Unglücklicherweise sorgen große Nachrichten in den Routern aber auch für lange Reservierungszeiträume der Ressourcen (Pufferspeicher, Belegung der Kreuzschiene und Ausgangsleitungen). Die Folge ist also eine erhöhte Wartezeit blockierter Nachrichten.

Zerlegt man das Paket weiter in sogenannte *Flits*, kann die Ressourcenverwaltung feingranularer erfolgen und die Ressourcen fairer verteilen. Auf Flits basierende Flusskontrollen sind das *Wormhole-* oder das *Virtual-Channel*-Verfahren.

2.1.3.1. Store-And-Forward

Mit der Store-And-Forward (SAF) Flusskontrolle werden die Pakete erst vollständig empfangen, bevor sie den Router über die entsprechende Ausgangsleitung verlassen können. Zusätzlich muss zuvor der nächste Router auf dem Pfad des Pakets seine Bereitschaft signalisieren, das Paket vollständig empfangen und speichern zu können [Dally und Towles 2003]. Falls die nötigen Ressourcen für die Weiterleitung zum nächsten Router nicht verfügbar sind, verbleibt das Paket im Eingangspuffer des aktuellen Routers und belegt ansonsten keine weiteren Ressourcen. Die SAF Flusskontrolle ist zwar sehr leichtgewichtig, doch darf der Paketkopf erst weiter gesendet werden, wenn das Paket vollständig empfangen und gespeichert wurde. Dies wirkt sich negativ auf die Latenzzeiten der Pakete aus. Ferner kann ein Pufferspeicher immer nur ein Paket aufnehmen. Das gilt auch dann, wenn die Pakete deutlich kleiner sind, als die Speicherkapazität des Puffers. Werden also häufig kleine Pakete übertragen, sinkt durch den statischen Ener-

gieverbauch der leeren Pufferzellen die Effizienz des gesamten Puffers [Kahng 2009; Sun 2012].

2.1.3.2. Cut-Through

Die (virtual) Cut-Through (VCT) Flusskontrolle ist dem SAF-Verfahren sehr ähnlich. Auch hier werden die Eingangspuffer und die Ausgangsleitung exklusiv für ein ganzes Paket reserviert. Es vermeidet jedoch die hohen Latenzzeiten des SAF-Verfahrens, in dem die Routing- und Arbiter-Einheit bereits direkt beim Erhalten des Paketkopfes mit der Kanalbildung beginnen. Dadurch kann das Paket schneller zum nächsten Router weitergeleitet werden und verkürzt somit die Wartezeit im Router. Ein prominentes Beispiel für den Einsatz von VCT im Prozessorentwurf ist Intel's *Single-Chip Cloud Computer* [Howard 2010; Salihundam 2010] Prozessor mit 48 Pentium Kernen.

Auf Paketgranularität basierende Flusskontroll-Verfahren zeigen Nachteile bei Reservierung auf Paketebene. Insbesondere wenn die Pakete unterschiedlich groß sind. Kleine Pakete belegen den Pufferspeicher ebenso exklusiv, wie große Pakete, welche die Pufferkapazität voll ausschöpfen. Dadurch sinkt die Effizienz des Pufferspeichers [Kahng 2009; Sun 2012], da ein Teil des Puffers leer bleibt aber dennoch Energie verbraucht. Des Weiteren sind die Latenzen durch Aufstauen der Pakete bei Staus deutlich höher als bei vergleichbaren Flusskontroll-Verfahren, die auf Flits basieren [Flich und Bertozzi 2010].

Das Nachrichtenformat im unteren Teil der Abbildung 2.3 zeigt die Zerlegung eines Pakets in Flits. Zu einem Paket gehört immer ein Flit-Kopf, der die Routing-Informationen und weitere Steuerinformation enthält. Zusätzlich kann ein Paket aus einer variablen Anzahl von Daten-Flits bestehen. Der Flit-Schwanz bildet das Ende des Pakets und wird verwendet um die reservierten Ressourcen beim Verlassen des Routers wieder freizugeben. Daten-Flits und der Schwanz sind jedoch optional. Sehr kurze Nachrichten können daher bereits aus einem einzelnen Flit-Kopf bestehen. Auf Flits basierende Flusskontroll-Verfahren sind die Verfahren *Wormhole* und *Virtual Channel*.

2.1.3.3. Wormhole

Obwohl das Wormhole-Verfahren in der Fachliteratur oft als Routing-Strategie bezeichnet wird, ist es tatsächlich ein Verfahren der Flusskontrolle. Beim Wormhole-Verfahren handelt es sich um eine Erweiterung des VCT [Dall'Osso 2003; Dielissen 2003; Hu und Marculescu 2004]. Die Kanalbildung wird hier jedoch auf Flit-Granularität durchgeführt. Auch hier wird der Flit-Kopf bereits zum nächsten Router übertragen, wenn die Routing-Entscheidung getroffen und die Reservierung der Ausgangsleitung abgeschlossen ist. Wird der Kopf des Flits blockiert, nimmt der Eingangspuffer so viele Flits auf, bis dieser voll ist. Sofern das gesamte Flit im Puffer gespeichert werden kann und noch

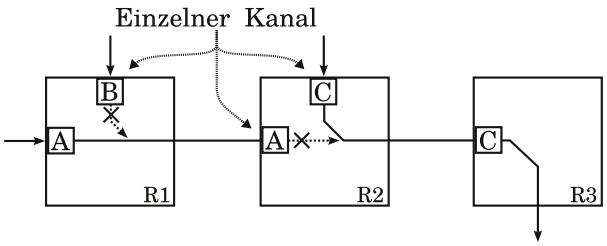


Abb. 2.4.: Die blockierte Nachricht A sorgt dafür, dass die Bandbreite des Kanals von Router 1 zu 2 nicht ausgenutzt werden kann.

weiterer Speicherplatz zur Verfügung steht, können auch weitere Flits von ihm aufgenommen werden[5]. Diese Art der Flusskontrolle wird beim Prozessor Tile64 [Wentzlaff 2007] von Tilera für die prozessorinterne Kommunikation auf dem Chip eingesetzt.

Häufig wird wegen des knappen Energie-Budgets jedoch nur wenig Pufferspeicher in den Routern implementiert [Dally und Towles 2003]. Das kann dazu führen, dass sich die Flits von längeren Paketen bei der Übertragung durchs Netz über mehrere Router erstrecken. Problematisch daran ist, dass die Verbindungsleitungen auch beim Wormhole-Verfahren exklusiv von einem ganzen Paket reserviert werden. Wird also ein Flit-Kopf blockiert, blockieren die nachfolgenden Daten-Flits die entsprechenden Ressourcen der anderen Router in denen sie sich aktuell befinden. Die Folge ist ein unproduktiver Leerlauf der reservierten Ressourcen. Abbildung 2.4 zeigt dies anhand eines Beispiels. Das Flit A wird am Router 2 durch die Verarbeitung von Flit C blockiert. Durch die kleinen Puffergrößen staut sich A über die Router R1 und R2 hinweg auf. Das Flit B wird in R1 blockiert, weil A weiterhin die Kanäle reserviert hält. Damit befindet sich die Verbindungsleitung von R1 zu R2 im Leerlauf und die Bandbreite bleibt ungenutzt, solang A blockiert ist.

2.1.3.4. Virtual-Channel

Bei dem Virtual-Channel (VC) Flusskontrollverfahren werden statt einem Kanal, mehrere virtuelle Kanäle auf eine physische Leitung abgebildet [Dally 1992; Dally und Towles 2003; Kavaldjiev 2004]. Dazu werden multiple Pufferspeicher genutzt, die jeweils einem virtuellem Kanal zugeordnet sind. Ist ein virtueller Kanal bereits von einem blockierten Flit belegt, sorgen Multiplexer dafür, dass ein weiteres Flit in einem anderen virtuellen Kanal aufgenommen wird. Zusätzlich erlaubt das VC-Verfahren dynamische Re-Allokationen von Ressourcen, sofern diese sich gerade im Leerlauf befinden. Die Bandbreite, welche beim Wormhole-Verfahren durch blockierte Flits ungenutzt blieb, kann damit besser ausgenutzt werden. In Anlehnung an das zuvor beim Wormhole-Verfahren verwendete Beispiel, zeigt Abbildung 2.5 das Beispiel diesmal für das VC-Verfahren. Auch hier wird das Flit A im Router 2 durch Flit C blockiert. Wo vormals jedoch das

[5]Eine spezielle Implementierung von [Matos 2011] ermöglicht den freien Speicherplatz anderer Eingangspuffer zu nutzen, wenn der eigentliche Eingangspuffer voll ist.

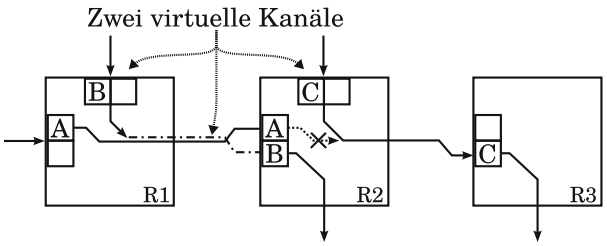


Abb. 2.5.: Mit Hilfe virtueller Kanäle kann Nachricht B trotz A direkt übermittelt werden.

Flit B im Router R1 blockiert wurde, kann B nun über einen freien virtuellen Kanal (grob gestrichelt) übertragen werden. Dazu wird der freie Kanal von R1 zu R2 genutzt und gleichzeitig die Bandbreite der Verbindungsleitung besser ausgenutzt.

2.1.4. Routing

Die Routing-Einheit implementiert Strategien, nach denen bestimmt wird, welche Route eine Nachricht vom Absender zum Empfänger durch das Netz nimmt. Die Strategien verwenden dazu spezielle Informationen aus dem Paket- bzw. Flit-Kopf und liefern mindestens eine Ausgangsleitung, um einen Kanal im Router herstellen zu können. Die Anzahl der zurückgelieferten Ausgangsleitungen hängt dabei von der Implementierung der Routing-Strategie ab. Eine gute Routing-Strategie verfolgt zwei Ziele [Dally und Towles 2003]:

(i) Die Last des Datentransfers möglichst ausbalanciert auf das Netz verteilen (*load balancing*). Insbesondere bei nicht gleichverteilten Sendemustern ist diese Fähigkeit wertvoll, denn nur wenn die Last auf das Netz ausbalanciert ist, kann die gesamte Bandbreite genutzt und das Maximum des Datendurchsatzes im Netz erreicht werden.

(ii) Die Pfade durch das Netz sollen sehr kurz sein. Je kürzer der Weg durch das Netz ist, desto weniger Verarbeitungsschritte sind nötig und entsprechend kleiner ist allgemein die Latenz eines Pakets/Flits.

Dabei ist es wichtig anzumerken, dass beide Ziele nicht immer miteinander vereinbar sind oder sich zum Teil sogar gegenseitig ausschließen können. Ein hoher Durchsatz im Netz lässt sich nur dann erreichen, wenn tatsächlich alle Verbindungsleitungen zu gleichen Teilen belastet werden. Erreicht wird dies dadurch, dass Nachrichten auf Pfaden mit wenig Last durch das Netz übertragen werden. Dadurch nimmt aber die Länge der Pfade durchs Netz im Durchschnitt zu und erhöht ebenfalls die durchschnittliche Latenz der Nachrichten. In [Duato 2002; Dally und Towles 2003] werden zwar Kompromisse vorgeschlagen, um dieses Problem zu entschärfen, jedoch gibt es bereits Ansätze, die durch intelligente Platzierung der kommunizierenden Programmteile positiv auf Engpässe im Netz einwirken [Schlingmann 2011; Dziurzanski und Maka 2013]. Beginnt man

also frühzeitig[6] den Datenstrom im Netz zu koordinieren (d.h. Platzierung der Sender und Empfänger), kann damit sowohl der Durchsatz, als auch die Latenz verbessert werden.

2.1.4.1. Routingklassen

Die Routing-Strategien lassen sich anhand vieler verschiedener Eigenschaften klassifizieren. Eine vollständige Diskussion dieser Eigenschaften wäre jedoch für diese Dissertationsarbeit zu umfangreich, weshalb im Folgenden nur die Kerneigenschaften angesprochen werden, die eine entsprechende Relevanz für diese Arbeit beinhalten. Erschöpfende Aufarbeitungen der Routing-Strategien können in [Duato 2002; Dally und Towles 2003; Flich und Bertozzi 2010] nachgeschlagen werden.

Die Kerneigenschaften der Routing-Strategien für diese Arbeit lassen sich zunächst darin unterschieden, wo die Routing-Entscheidungen getroffen werden; *verteiltes Routing* und *Quell-Routing* [Dall'Osso 2003; Kavaldjiev 2004]. Beim verteilten Routing entscheidet jeder Router für sich allein, welche Ausgangsleitung mit einem Eingangspuffer verbunden wird [Dally und Towles 2003]. Dafür nutzt ein Router die Zieladdresse der Nachricht aus dem Nachrichtenkopf und seine eigene lokale Adresse. Informationen über den restlichen Netzzustand, wie die aktuelle Belastung einzelner Router oder der Zustand der Verbindungleitung, sind für die Entscheidungsfindung nicht zwingend notwendig. Die Route durch das Netz ist also das Resultat der einzelnen Entscheidungen der verwendeten Router.

Analog dazu wird beim Quell-Routing bereits vor dem Absenden der Nachricht in den Kopf der Nachricht eingetragen, welche Route verwendet werden soll. Im Kopf der Nachricht befindet sich also eine Wegbeschreibung, nach der die Router die Nachricht weiterleiten. Um die Wegbeschreibung anfertigen zu können, benötigt das Quell-Routing allerdings genaues Wissen über die Topologie des Netzes. Dies ist besonders dann von Vorteil, wenn die Topologie nicht gleichmäßig aufgebaut ist und die Unregelmäßigkeiten global bekannt sind.

Die Implementierung dieser beiden Klassen kann weiter zwischen *Look-Up Tabellen* und *endlichen Automaten* unterschieden werden [Dally und Towles 2003]. In den Look-Up Tabellen sind bereits im Voraus berechnete Routen für alle möglichen Empfänger hinterlegt. Der Vorteil daran ist, dass die Routen dadurch sehr genau geplant werden können. Das ermöglicht einen hohen Datendurchsatz im Netz. Der Nachteil ist jedoch, der große Speicheraufwand für die Tabellen, die entweder im Router (beim verteilten Routing) oder beim Absender (bei Quell-Routing) vorgehalten werden müssen. Der Speicherbedarf jeder einzelnen Tabelle wächst dabei linear mit der Anzahl der Netzteilnehmer. Endliche Automaten können dagegen sehr einfach implementierte Hardware-Module sein [Duato 2002]. Besonders dann, wenn diese Module

[6] In diesem Fall auf Betriebssystem-Ebene

(i) lokal in jedem Router identisch implementiert sind

(ii) und bei zwei-dimensionalen Topologien, wie dem 2D-Gitter, eingesetzt werden.

Zusätzlich kann das auf endlichen Automaten basierende Routing reaktiv auf Änderungen des lokalen Routers eingehen. Look-Up Tabellen benötigen dafür erst eine aktualisierte Version des Netzzustandes.

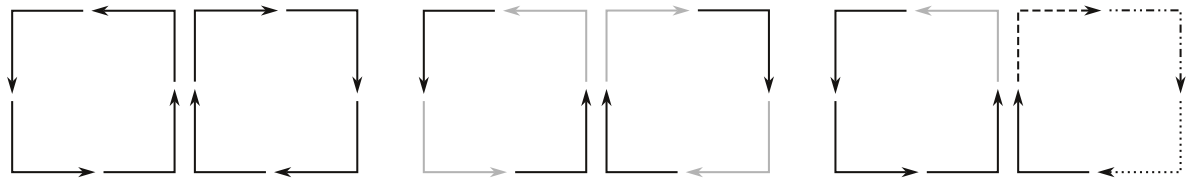
Abschließend wird weiter unterschieden, ob die Routen zweier Kommunikationspartner durch das Netz stets gleich oder variabel sind. Beim *deterministischen* Routing [Benini und De Micheli 2002; Dally und Towles 2003; Goossens 2005] sind die Routen stets gleich und damit vorhersagbar. Bei diesen Strategien wird von Router zu Router nur lokales Wissen über das Netz in der immer gleichen Prozedur verarbeitet. Der globale Zustand des Netzes wird also bei der Entscheidungsfindung nicht berücksichtigt und alternative Routen werden nicht wahrgenommen. Zusätzlich muss jede Routing-Entscheidung dazu führen, dass die Nachricht pro Routing-Schritt dem Ziel tatsächlich näher kommt (*minimale Pfadlänge*). Beliebige Richtungsänderungen sind nicht erlaubt. Damit werden sowohl Deadlocks als auch Livelocks verhindert.

Adaptive Routing-Strategien nutzen Informationen über den globalen Zustand des Netzes. Mit Hilfe alternativer Routen durch das Netz, können überlastete Bereiche umgangen oder defekte Verbindungsleitungen gemieden werden. Zusätzlich bestehen die adaptiven Routing-Strategien aus zwei Modulen. Das erste Routing-Modul liefert eine Liste mit möglichen Kandidaten (Ausgangsleitungen) und ein Selektionsmodul entscheidet anhand der Zustandsinformation, welche der Ausgangsleitungen mit dem Eingangspuffer verbunden werden soll. Bei der Selektion wird aber nicht zwingend die Regel der minimalen Pfadlänge eingehalten, was zu einem Deadlock, bzw. Livelock führen kann. Wie Deadlocks und Livelocks dennoch vermieden werden können wird im nächsten Abschnitt erläutert.

Die hohe Flexibilität der adaptiven Strategien erfordert einen intensiven Informationsaustausch zwischen den Routern, die die Routing-Strategie implementieren. Die Folge ist ein erhöhter Aufwand, um die Zustandsinformationen im Netz zu verteilen. Alternativ dazu gibt es die *partiell* adaptiven Routing-Strategien. Hier werden nur die Zustände der direkt benachbarten Router abgefragt und mit in die Routing-Entscheidung einbezogen. Liegen keine Informationen über die Nachbarn vor (alles ist fehlerfrei) wird nach dem deterministischen Verfahren die Kanalbildung vollzogen. Liegen Zustandsinformationen vor, werden Kandidaten ermittelt und das Selektionsmodul wählt auf Grundlage der Zustandsinformationen den besten Kandidaten für den Kanal aus.

2.1.4.2. Deadlocks und Livelocks

Da (partiell) adaptive Routing-Strategien nicht an die Regel der minimalen Pfade gebunden sind, können diese Strategien zu Blockaden führen, bei denen das gesamte Netz keine Nachrichten mehr verarbeiten kann. Dies wird als *Deadlock* bezeichnet. Hierfür reichen



(a) Zyklische Abhängigkeit die zu Deadlocks führen.

(b) Gültige (schwarze Pfeile) und ungültige (graue Pfeile) Richtungsänderungen der XY-Strategie.

(c) Maximal erlaubte Richtungswechsel für eine Deadlock-freie 2D-Gitter Topologie.

Abb. 2.6.: Zyklische Abhängigkeitsverhältnisse (a) und deren Auflösung durch dimensionsorientierte Routing-Strategien (b) oder Turn Modelle (c) [Glass und Ni 1992].

bereits wenige dauerhaft blockierte Ressourcen, um Rückstaus zu bilden, die das gesamte Netz stilllegen [Duato 1993; Dally und Towles 2003]. Der Grund für ein Deadlock sind zyklische Abhängigkeiten von Nachrichten, welche bei der Reservierung von Netzressourcen entstehen können. Dies kann vor allem bei adaptiven Routing-Strategien auftreten, da in diesem Fall für Nachrichten keine Beschränkungen bezüglich der Senderichtung gelten und damit Nachrichtenpfade geschlossene Kreise bilden können. Abbildung 2.6(a) zeigt alle im Netz möglichen Richtungswechsel in einer 2D-Gitter Topologie. Betrachtet man die Pfeile als Abhängigkeiten, erkennt man ein zyklisches Abhängigkeitsverhältnis. Nachrichten die in so einer Konstellation im Netz aufeinander treffen, können sich also gegenseitig blockieren, was zu einem Deadlock führt.

Um Deadlocks im Netz zu vermeiden, gibt es zwei Herangehensweisen. Zum einen können bereits beim Entwurf der Routing-Strategie Deadlocks verhindert werden, indem die Routing-Strategie von vorneherein zyklische Abhängigkeiten in der gegebenen Netztopologie verhindert. Die einfachste und gleichzeitig restriktivste Methode ist die dimensionsorientierte Routing-Strategie. Nachrichten, die mit diesen Strategien verarbeitet werden, dürfen immer nur in eine Richtung pro Dimension übertragen werden. Bei der XY-Strategie [Ni und McKinley 1993] einer 2D-Gitter Topologie werden Nachrichten zunächst immer nur in auf- oder absteigende X-Richtung übertragen. Eine Richtungsänderung in Y-Richtung ist erst dann möglich, wenn sich die Nachricht an der korrekten Position in X-Richtung befindet. Erst dann wird die Nachricht in Y-Richtung weiter übertragen. Das daraus entstehende Abhängigkeitsverhältnis ist in Abbildung 2.6(b) eingezeichnet. Die gültigen Richtungswechsel sind durch die schwarzen Pfeile und die ungültigen durch die grauen Pfeile angedeutet. Da aus den gültigen Richtungswechseln keine zyklischen Abhängigkeiten erzeugt werden können, ist diese Routing-Strategie per Entwurf frei von Deadlocks.

Die (partiell) adaptiven Routing-Strategien wären jedoch mit dem eben beschriebenen dimensions-orientierten Verfahren in ihrer Flexibilität stark eingeschränkt. Oft sind nur wenige ungültige Richtungswechsel notwendig, um Deadlock-Freiheit zu garantieren. Die Art in der diese ungültigen Richtungswechsel gewählt werden, wird als *turn model* be-

zeichnet und ermöglicht adaptiven Routing-Strategien mehr Flexibilität [Glass und Ni 1992]. In Abbildung 2.6(c) wurde zunächst nur der Richtungswechsel *Nord-West* (grauer Pfeil, linker Zyklus) als ungültig definiert. Für den anderen Kreis ergeben sich damit drei Möglichkeiten (gestrichelte Pfeile) um jeweils einen weiteren Richtungswechsel für ungültig zu erklären und damit zyklische Abhängigkeiten zu verhindern. Der vierte Richtungswechsel (durchgezogener Pfeil) ist als ungültiger Richtungswechsel nicht erlaubt, da man aus den dann verbleibenden Richtungswechseln einen Zyklus in Form der Ziffer Acht erzeugen könnte.

Die zweite Möglichkeit, um im Netz Deadlock-Freiheit zu garantieren, beruht auf dem Einsatz von speziellen virtuellen Kanälen. Diese virtuellen Kanäle werden als *Ausbruch-Kanal* (escape channel) bezeichnet, und können nur dann verwendet werden, wenn alle anderen Kanäle eines Routers blockiert sind [Duato 1995; Duato 1996]. Eine blockierte Nachricht kann im Fall eines Deadlocks exklusiv so einen Ausbruch-Kanal nutzen, um, an den anderen Nachrichten vorbei, übertragen zu werden. Beim Verlassen des Routers gibt die ausbrechende Nachricht ihre Ressourcen wieder frei und der Deadlock wird durchbrochen. Zwar bewahrt man damit einer (partiell) adaptive Routing-Strategie die volle Flexibilität, jedoch erkauft man sich dies durch zusätzliche Puffer-Speicher. Da diese Puffer-Speicher nur im Notfall verwendet werden dürfen, kann ansonsten kein weiterer Vorteil in Bezug auf Durchsatz oder Latenz gewonnen werden, obwohl sie dennoch zum statischen Energie- und Platzverbrauch des Netzes beitragen.

2.2. Fehlertoleranz im NoC

Das prozessorinterne Kommunikationsnetz wird unter den Folgen der immer kleiner werdenden Strukturgrößen, mit denen die zukünftigen Prozessoren gefertigt werden, zu leiden haben. Wie in der Einführung bereits beschrieben, wird es immer schwieriger werden, nach der Fertigstellung eines Prozessors seine Fehlerfreiheit zu garantieren. Zusätzlich wirken während des Betriebs physikalische Effekte auf den Prozessor ein, sodass Alterungsprozesse zu erwarten sind. Aus heutiger Sicht müssen also Fehlproduktionen oder auftretende Fehler im laufenden Betrieb beim Entwurf eines Prozessors berücksichtigt werden. Da es nicht realistisch ist alle Fehler in einem Prozessor zu vermeiden, bemüht sich die Forschung bereits seit längerem um Verfahren, mit denen Fehler in einem Prozessor tolerierbar werden.

Die Fehlertoleranz in Kommunikationsnetzen ist ein Teilbereich der fehlertoleranten Systeme und wird bereits breit beforscht. Die Fachliteratur bietet daher eine Reihe an verschiedensten Konzepten an, wie die Zuverlässigkeit zukünftiger Prozessorgenerationen verbessert werden kann. Diese Konzepte basieren auf Fehlermodellen, die das Fehlerverhalten beschreiben. Die Konzepte selbst bestehen im Wesentlichen wiederum aus zwei Teilbereichen. Der erste Bereich widmet sich der Fragestellung, wie ein Fehler im Netz identifiziert werden kann (Fehlererkennung). Die Fehlerbehebung stellt den zweiten Bereich dar. Hier wurden Konzepte entwickelt, um die Ausbreitung eines erkannten Fehlers zu verhindern und Auswirkungen der Fehler auf das System zu minimieren oder ganz zu vermeiden.

2.2.1. Fehlermodell

Neben einem detaillierten Modell des zugrundeliegenden Kommunikationsnetzes, ist für den Entwurf fehlertoleranter Systeme vor allem das Fehlermodell entscheidend. Nur wenn bekannt ist, mit welcher Art von Fehlern im Netz gerechnet werden muss, lassen sich effektive Verfahren zur Fehlertoleranz entwickeln. Dazu gehört auch eine Einschätzung, welche Bereiche des Netzes von den modellierten Fehlern betroffen sind und welche nicht. Die folgenden beiden Unterabschnitte definieren dazu zunächst die möglichen Fehlerarten und anschließend die kritischen Pfade im Netz, die gegen Fehler abzusichern sind.

2.2.1.1. Fehlerarten

Innerhalb dieser Arbeit werden Fehler des Netzes durch ihre zeitliche Dauer unterschieden. Hierfür werden Fehler in drei Kategorien aufgeteilt: *Transient*, *intermittierend* und *permanent*.

Transiente Fehler definieren sich durch ihren sehr kurzlebigen Effekt, den sie auf ein Bauteil eines Systems ausüben. Oft führen einzelne und zufällige Ereignisse, wie Einschläge von Neutronen oder Alpha-Teilchen in eine Speicherzelle, zu sogenannten Bitfehlern[7]. Wie später noch erläutert wird, können Bitfehler leicht erkannt und zum Teil sogar an Ort und Stelle korrigiert werden.

Permanente Fehler stellen das genaue Gegenteil zu transienten Fehlern dar. Diese Fehlerart entsteht häufig durch *Stuck-At*-Fehler. Die Ausgangsleitung einer logischen Einheit bleibt bei dieser Fehlerart, unabhängig vom Eingabewert, konstant auf einem fixen Wert. Die Gründe für so einen Effekt sind vielseitig:

- Durch Materialwanderung verursachte Zerstörung der Verbindungsleitung (*Elektromigration*, Stuck-At 0).
- Durch Ablagerung leitendem Materials verursachter Kurzschluss (*Oxide Breakdown*, Stuck-At 1).
- Verschiebung von Schwellwerten, bei denen ein Transistor schaltet (*Hot Carrier Injection*, kann zu beiden Stuck-At Fehlern führen).

Verstärkt werden diese Alterungsprozesse durch hohe thermale Belastungen.

Intermittierende Fehler sind häufig die Vorboten von permanenten Fehlern. Das hängt vor allem mit den oben genannten Alterungsprozessen zusammen. Zunächst sorgt das Altern dafür, dass Fehler sporadisch in einem Transistor oder Verbindungsleitung auftreten. Ob der Fehler im Transistor zum fehlerhaften Verhalten führt oder nicht, hängt häufig zusätzlich von der thermalen Belastung ab. Die Folge in diesem Stadium ist, dass bei hohen Temperaturen ein Fehler zu einem Stuck-At-Fehler führt und bei niedrigen Temperaturen einen fehlerfreien Betrieb ermöglicht. Da die Temperaturen aber schwanken können, tritt das Fehlverhalten nur zwischenzeitlich auf. Mit dem Voranschreiten des Alterungsprozesses können aus intermittierenden Fehlern permanente werden.

2.2.1.2. Verwundbare Punkte des Netzes

Für eine effektive Strategie der Fehlererkennung und der späteren Behebung des Fehlers, ist es notwendig die Orte, an denen die Fehler entstehen können, zu identifizieren. Dazu wird in dieser Arbeit zwischen den Verbindungsleitungen, dem Router und dem Prozessorkern unterschieden.

Verbindungsleitungen (Datenpfad) beinhalten keine Kontrolllogik und stellen damit ausschließlich den kritischen Pfad der Daten dar. Fehlererkennung und Fehlerbehebung

[7] Veränderung einzelner Bits in einem Datum.

beschränken sich daher auf die Auswertung der gesendeten Daten, die über die überwachte Verbindungsleitung übertragen wurde. Dazu wird, wie später genauer beschrieben, die Integrität der Daten an bestimmten Positionen im Netz geprüft. Die Orte, an denen die Prüfung vollzogen wird, werden so gewählt, dass die betroffene Verbindungsleitung direkt ermittelbar ist. Als Konsequenz eines ermittelten Fehlers wird die fehlerhafte Verbindungsleitung abgeschaltet oder durch eine Ersatzleitung ersetzt.

Der Router (Daten-/Kontrollpfad) beinhaltet mit den verschiedenen Funktionseinheiten neben dem kritischen Datenpfad auch den kritischen Kontrollpfad. Fehlerhafte Funktionseinheiten, wie die Routing- und Arbiter-Einheit oder der Kreuzschiene, können dazu führen, dass Nachrichten über eine falsche Ausgangsleitung den Router verlassen. Das ist kritisch, da solche Fehler zu Deadlocks im Netz führen können. Da dieses Fehlverhalten nicht direkt anhand der gesendeten Nachricht (also dem Datenpfad) erkennbar ist, müssen die entsprechenden Funktionseinheiten des Routers mit speziellen Verfahren zur Fehlererkennung überwacht werden.

Der Prozessorkern ist der Beginn und das Ende des Datenpfades und daher Teil des Fehlermodels. Zwar kann die Fehlererkennung durch kerninterne Mechanismen zum Teil sehr genau kerninterne Fehler identifizieren, jedoch ist dies ein Teil der Prozessorkernüberwachung und steht damit nicht unmittelbar im Zusammenhang mit der Fehlerlokalisierung im Netz. Einzig der Ausfall des gesamten Kerns hat Auswirkung auf das Lokalisierungsverfahren, da in so einem Fall keine Nachrichten abgesendet werden. Überwacht wird so ein Ausfall beispielsweise durch eine (de)zentrale Überwachungseinheit, die in bestimmten Abständen von den Prozessorkernen eine Statusnachricht erwartet.

2.2.2. Fehlererkennung und Behebung

Die Fehlererkennung ist der erste Schritt um ein System gegen die Auswirkungen von Fehlern abzusichern. Hierzu gibt es in der Literatur ein breites Spektrum an verschiedenen Verfahren, wie Fehler auf dem Daten- und Kontrollpfad des Netzes effektiv und effizient erkannt werden können. Die folgende Diskussion der Ergebnisse aus verwandten Forschungsarbeiten ist in zwei Bereiche unterteilt und unterscheidet die jeweiligen Orte, an dem die Fehlererkennung durchgeführt wird. Die Unterscheidung findet zwischen den Konzepten *Router-zu-Router* (R2R) und *Ende-zu-Ende* (E2E) statt.

2.2.2.1. Router-zu-Router Fehlererkennung

Das Router-zu-Router (R2R) Konzept umfasst sowohl den Datenpfad als auch den Kontrollpfad des Kommunikationsnetzes. Wie der Name bereits suggeriert, werden die Fehlererkennungsverfahren von jedem einzelnen Router im Netz durchgeführt. Die bisher

entwickelten Verfahren zur Fehlererkennung, bestehen aus dem Ausnutzen der *Redundanz* (Informationsredundanz, zeitliche oder räumliche Redundanz), *Selbsttest-Routinen* (build-in selft test, BIST) oder der *regelbasierenden* Fehlererkennung.

Anwendungsnachrichten können verwendet werden, um die Verbindungsleitungen im Netz zu testen. Dazu werden die Nachrichten in redundanter Form übermittelt. Zum einen kann die Redundanz durch zusätzliche Informationen in den Nachrichten erzeugt werden, oder zum anderen durch mehrfaches Senden der gleichen Nachricht.

Bei den redundanten Informationen werden die Nachrichten vor dem Absenden mit speziellen Bits kodiert[8]. Der Empfänger prüft die Nachricht anhand der redundanten Bits der Kodierung und entfernt diese nach der Prüfung wieder aus der Nachricht. Abhängig von der Art der Informationsbits, kann der Empfänger eine beschränkte Anzahl an Bitfehlern direkt in der Nachricht korrigieren (forward error correction, FEC). Sind mehr Bits in der Nachricht fehlerhaft, kann das zwar erkannt werden, jedoch muss die Nachricht wie in [Murali 2005] vom Sender erneut übertragen werden (backward error correcting, BEC), um die Fehler in der Nachricht zu korrigieren.

Bei der zeitlichen Redundanz wird die Nachricht mehrfach aber zeitlich versetzt übertragen. Der Empfänger vergleicht nach Erhalt beider Nachrichten, ob das Original mit dessen Dublette identisch ist. Sind die Nachrichten unterschiedlich, trat während der Übertragung ein Fehler auf und eine dritte Nachricht ist notwendig, um die korrekte Nachricht bestimmen zu können.

Die zeitliche Redundanz hat dabei einen entscheidenden Nachteil. Während transiente Fehler in der Regel nur eine der beiden Nachrichten verändert, sind von einem permanenten Fehler beide Nachrichten gleichermaßen betroffen. Das bedeutet, der Fehler steckt in beiden Nachrichten und der Vergleich des Empfängers stellt keinen Unterschied fest. Datenredundanz ist daher für R2R-Verfahren nur bei transienten Fehlern hilfreich.

Die Autoren aus [Park 2006; Grecu 2006b; Fick 2009] erweiterten das oben beschriebene hybride Verfahren. Statt an den Ein- und Ausgängen des Routers den Nachrichteninhalt auf Korrektheit zu prüfen, wird vorgeschlagen, die Prüflogik auch innerhalb des Routers einzusetzen. In [Park 2006] wird jede Funktionseinheit des Routers mit ECC-Logik erweitert. Ein Fehler in der Nachricht kann damit präzise erkannt und lokalisiert werden. [Fick 2009] erweitert hingegen nur die Puffer durch ECC-Logik. Die Funktionseinheiten des Routers werden mit BIST-Logik getestet. Damit ist nicht nur der Inhalt der Nachricht gesichert. Es wird ferner sichergestellt, dass eine Nachricht nicht durch einen Fehler in den Funktionseinheiten des Routers auf eine falsche Ausgangsleitung gelegt wird.

Bei den BIST-Verfahren handelt es sich um Hardware-Module im Router, die mit Hilfe von speziell präparierten Nachrichten oder Signalen, die korrekte Funktionsweise der Daten- und Kontrollpfade testen. Die Tests werden dazu in regelmäßigen Abständen

[8] Die Kodierungen basieren meist auf CRC-verfahren oder Hemming-Codes.

durchgeführt, sofern die zu testende Komponente nicht gerade verwendet wird [Nicolaidis und Zorian 1998].

In [Pande 2005a; Grecu 2006a] werden BIST-Verfahren präsentiert, bei denen die Verbindungsleitungen zwischen zwei Routern mit Hilfe von speziellen Signalmustern getestet werden. Ziel ist es, Fehler in der Verbindungsleitung durch Nebensprechen der Signalleitung zu ermitteln. Dazu verfügen zwei benachbarte Router an den Ausgangsleitungen jeweils über einen *Test Data Generator* (TDG) und an den Eingangsleitungen jeweils einen *Test Error Detector* (TED). Das TDG-Modul erzeugt die Testnachrichten, die beiden Modulen bekannt sind. Empfängt das TED-Modul eine Testnachricht, wird diese auf Korrektheit geprüft. Fehler in der Testnachricht lassen sich präzise auf die einzelnen Signalleitungen der Verbindungsleitung abbilden, sodass die fehlerhafte Komponente exakt bestimmt werden kann.

Ein Nachteil der BIST-Verfahren besteht darin, dass die Testnachrichten mit den Anwendungsnachrichten um den Zugriff auf die Netzressourcen konkurrieren. Dies bedeutet, dass bei hoher Auslastung des Netzes, die Anwendungsnachrichten entweder blockiert werden könnten, oder die Testmodule lange warten müssen, bis eine entsprechende Leitung *frei* ist, um diese zu testen.

Das Verfahren von [Prodromou 2012] stellt eine regelbasierende Fehlererkennung vor und überwacht ausschließlich den Kontrollpfad eines Routers. Hierzu werden Regeln aller Funktionseinheiten ermittelt, die jeweils die gültigen Ausgaben der Funktionseinheit beschreiben. Die Regeln werden dann in spezielle Prüfmodule programmiert. Zusätzlich können Funktionseinheiten danach zusammengefasst werden, um im Verbund neue Regeln zu erzeugen. Alle Regeln zusammen sind dann eine Menge gültiger Zustände, in denen sich ein korrekt funktionierender Router befinden darf. Sobald eine Funktionseinheit eine Ausgabe produziert, wird diese auf die entsprechenden Regeln angewendet. Liegt dann ein Regelverstoß vor, muss ein Fehler in den Funktionseinheiten aufgetreten sein, die mit dieser Regel assoziiert ist. Auf diese Weise lassen sich Fehler bis in die Router-Pipeline hinein zurückverfolgen und lokalisieren.

2.2.2.2. Ende-zu-Ende Fehlererkennung

Bei diesem Konzept wird eine Nachricht erst beim Empfangskern auf Korrektheit geprüft. Die Fehlererkennung wird somit von den Routern auf die kommunizierenden Prozessorkerne verschoben. Ohne die Überprüfung der Nachrichten in den Routern, sind dort auch keine speziellen Puffer nötig, um die Kopie der Nachricht für eine erneute Übermittlung bereitzuhalten. Die damit geringeren Kosten durch Pufferspeicher in den Routern macht die Ende-zu-Ende Fehlererkennung attraktiv.

In [Cota 2008] wird das Verfahren von [Grecu 2006a] (siehe oben, TDG und TED-Module) erweitert, in dem die Testmodule in den Prozessorkern eingebettet sind. Damit werden ebenfalls die Verbindungsleitungen zwischen Router und Prozessorkern über-

wacht. Neben dem Prüfen der Signatur einer Nachricht, wird bei [Cota 2008] die Funktionalität der Router mitberücksichtigt. Wenn ein Empfänger eine Testnachricht erwartet, beginnt ein spezieller Time-Out-Zähler mit einer Wartefrist. Ist dieser abgelaufen, kann geschlussfolgert werden, dass die Nachricht während des Übertragens wegen eines Fehlers

(i) verworfen werden musste oder

(ii) auf eine falsche Route geleitet wurde.

Ein ähnliches Verfahren wird von [Abdel-Khalek 2011] vorgeschlagen. Auch hier wird eine Signatur der Nachricht vom Absender berechnet. Jedoch werden diese nicht an die Nachricht geheftet. Stattdessen wird die Signatur zusammen mit einer eindeutigen ID über ein separates Netz vorausgeschickt (look-ahead-signature). Der Empfängerkern nimmt die Signatur entgegen und speichert diese für den späteren Abgleich mit der Nachricht ab. Wird dann die eigentliche Nachricht empfangen, berechnet der Empfänger ebenfalls die Signatur der Nachricht und vergleicht diese mit den bereits aus dem separaten Netz empfangenen Signaturen.

Alle bisher beschriebenen Verfahren (sowohl R2R als auch E2E) haben gemeinsam, dass die gesammelten Fehlerinformationen nur lokal genutzt werden. Bei den R2R-Verfahren haben nur die jeweiligen Router Kenntnis über den eigenen Zustand. Es findet jedoch kein Austausch von Informationen mit einer höheren Verwaltungsinstanz wie dem Betriebssystem statt. Wie in [Schlingmann 2011; Dziurzanski und Maka 2013] beschrieben wird, sind gerade Informationen über den Zustand des Netzes kritisch für eine intelligente Prozessverteilung auf dem Prozessor.

2.2.3. Fehlerlokalisierung

Weit weniger Arbeiten finden sich zu den Lokalisierungsverfahren für ein globales Wissen über den Zustand des Netzes. Aus den verschiedenen zuvor erwähnten Verfahren können auch Lokalisierungsverfahren abgeleitet werden. Gerade die R2R-Konzepte, eignen sich sehr gut, um einen entstandenen Fehler einer Funktionseinheit zuzuordnen. Das zum Teil engmaschige Überwachen der Funktionseinheiten, wie in [Park 2006; Grecu 2006b; Fick 2009], ermöglicht eine präzise Lokalisierung.

Dennoch verbleiben diese Fehlerinformationen stets lokal und werden nur zum Abschalten der Verbindungsleitung (inklusive Anpassung der Routing-Strategie) oder des gesamten Routers genutzt. Eine (de-)zentrale Einheit zur Prozessverwaltung von Programmteilen ist jedoch auf diese Information angewiesen. Der naivste Ansatz, den Router selbst entsprechende Statusnachrichten erzeugen zu lassen, ist unattraktiv, da das nur mit Hilfe weiterer Hardware im Router zu realisieren ist.

Der erste Versuch, den Spalt zwischen der lokalen Fehlerinformation und dem globalen Netzstatus zu schließen, wurde von [Ghiribaldi 2011] vorgeschlagen. Die lokalen Infor-

mationen werden bei jedem Boot-Prozess des Prozessors durch ein BIST-Verfahren der Router erzeugt. Die dabei entstehenden lokalen Datensätze werden in Statusnachrichten verpackt und über ein separates Netz an eine zentrale Verwaltungseinheit geschickt. Dort werden die Datensätze der Router ausgewertet. Im Falle eines Fehlers, wird ein spezieller Konfigurationsdatensatz von der Verwaltungseinheit erzeugt [Ghiribaldi 2013] und an alle Router über das separate Netz übertragen. Die Konfiguration beinhaltet Routing-Informationen, welche verhindern sollen, dass durch den Fehler Deadlocks entstehen können. Worauf das Verfahren jedoch nicht eingeht, sind permanente Fehler, die während der Laufzeit des Prozessors auftreten. Da die Prüfung des Netzes nur Teil des Boot-Vorgangs ist, können permanente Fehler während der Laufzeit demnach nicht erkannt werden. Außerdem beruht dieser Ansatz auf der Verwendung eines separaten Netzes und erweitert zusätzlich die Router um die Fähigkeit selbstständig Nachrichten zu erzeugen.

Die Arbeiten von [Shamshiri 2011; Ghofrani 2012] verwenden dagegen auch Informationen des Netzes, die während der Laufzeit erhoben werden. Beide Arbeiten setzen jedoch auf Auswertungen von Anwendungsnachrichten basierend auf deren Signaturen auf. Dazu wird in [Shamshiri 2011] ein ECC-Verfahren verwendet, bei dem bis zu vier Bitfehler direkt korrigiert werden können. Zusätzlich wird bei einer Nachricht mit multiplen Bitfehlern geprüft, ob sich diese Fehler

(i) auf verschiedene Flits aufteilen und

(ii) ob die Position der gekippten Bits in den Flits jeweils gleich ist.

Falls die Position gleich ist, kann daraus geschlossen werden, dass ein permanenter Fehler auf der Route der fehlerhaften Nachricht liegt. Die Prozessorkerne senden in diesem Fall eine Statusnachricht an eine zentrale Verwaltungseinheit. Diese vermerkt alle Verbindungsleitungen, die an der Übertragung der fehlerhaften Nachricht beteiligt waren. Nach einer nicht näher definierten Zeit werden die vermerkten Verbindungsleitungen durch die Verwaltungseinheit ausgewertet und die Position des Fehlers bestimmt.

Das Verfahren von [Ghofrani 2012] hingegen unterscheidet sich dadurch, dass die Verwaltungseinheit mit Hilfe von Wahrscheinlichkeiten (basierend auf der verwendeten Routing-Strategie) die Position des Fehlers schätzt. Der Vorteil dieses Verfahrens liegt darin, dass auch nicht-deterministische Routing-Strategien unterstützt werden und dadurch das Gesamtsystem robuster gegen permanente Fehler ist. Dennoch beinhalten beide Ansätze eine Schwäche bezüglich permanenter Fehler am Randbereich des Prozessors. Statistisch ist im Zentrum des prozessorinternen Netzes die Belastung höher als an dessen Rändern [Duato 2002]. Um jedoch eine möglichst eindeutige Aussage über die Position des Fehlers treffen zu können, müssen möglichst viele Nachrichten unterschiedlicher Sender-Empfänger-Paare bei der Verwaltungseinheit mit diesem Fehler assoziiert werden[9].

[9] Diese Schwäche und eine entsprechende Lösung werden im Abschnitt 4.4.4 detailliert diskutiert.

Eine andere Herangehensweise präsentiert [Schley 2013]. Hier wird ein vollständig dezentrales Verfahren vorgeschlagen um schrittweise die Position eines permanenten Fehlers einzugrenzen. Die zugrundeliegende Idee ist, Nachrichten im Stil eines E2E-Verfahrens auf Fehler zu prüfen und mit einer Bestätigungsnachricht (ACK oder NACK) zu quittieren. Wird eine Nachricht nicht fehlerfrei übermittelt oder geht diese wegen eines Fehlers auf dem Weg zum Ziel verloren, wird auf der Hälfte des Pfades ein sogenannter *Zwischenhalt* als Relay eingesetzt[10]. Kann die Nachricht im zweiten Versuch ebenfalls nicht bis zum Zwischenhalt übertragen werden, wird die Wegstrecke abermals halbiert und ein neuer Zwischenhalt bestimmt. Dies wird solang fortgesetzt, bist der erste Zwischenhalt den Empfang einer korrekten Nachricht quittiert. Durch das stetige Halbieren der Wegstrecke kann der permanente Fehler im Netz sukzessive eingegrenzt werden[11]. Problematisch jedoch ist, dass dieses Verfahren ausschließlich mit deterministischen Routing-Strategien funktioniert. Ferner müssen die Kerne, welche als Zwischenhalt dienen, die empfangene Nachricht verarbeiten und sind damit für den Empfang eigener Nachrichten kurzzeitig blockiert. Zusätzlich werden laufende Anwendungen auf dem Kern pausiert, um die Nachricht zu verarbeiten.

2.3. Die Teraflux Basisarchitektur

Es wird erwartet, dass die grundlegenden Hardware-Blöcke, wie Prozessorkerne und das Kommunikationsnetz, aufgrund der immer kleiner werdenden Strukturgrößen an Zuverlässigkeit einbüßen. Eine große Herausforderung ist es daher, aus unzuverlässiger Hardware ein zuverlässiges System zu entwerfen.

In diesem Abschnitt wird die Prozessorarchitektur und die darin enthaltene Fehlererkennungseinheit (Fault Detection Unit, FDU) aus dem Forschungsprojekt TERAFLUX beschrieben. Die Einführung in die Prozessorarchitektur findet bereits an dieser Stelle statt, da die nachfolgenden Kapitel auf diese aufsetzen werden. Die Entwicklung dieser Architektur beruht, sofern nicht anders beschrieben, auf den Arbeiten der Konsortiumspartner des TERAFLUX-Projektes.

2.3.1. Allgemeine Architekturbeschreibung

Die TERAFLUX-Architektur wird, aus Sicht der Fehlertoleranz und der Kommunikationsnetze, in zwei Bereiche unterteilt. Wie aus Abbildung 2.7 hervorgeht, wird ein Teil der Prozessorkerne für ein Service-Cluster reserviert, der das Betriebssystem ausführt. Das Betriebssystem verwaltet unter anderem die Platzierung der Prozesse auf dem gesamten Prozessor [Schlingmann 2011].

[10] Hat die Nachricht den Zwischenhalt ohne Fehler erreich können, wird dieser als Checkpoint für eine erneute Übertragung der Nachricht verwendet.

[11] Ähnlich wie die Suche in einem binären Baum.

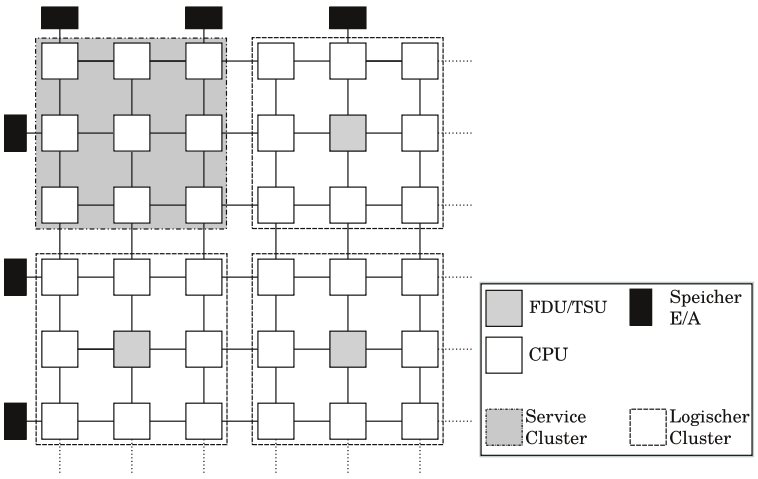


Abb. 2.7.: Schematischer Aufbau der Prozessor-Architektur aus der Sicht der Fehlertoleranz.

Die restlichen Bereiche des Prozessors werden aus einer Reihe *logischer Cluster* gebildet. Ein Cluster besteht aus einer bestimmten Anzahl an überwachten Prozessorkernen und zwei Verwaltungseinheiten. Die *Thread Scheduling Unit* (TSU) ist für die Prozessverwaltung innerhalb eines Clusters zuständig und verteilt die Arbeitspakete, die vom Betriebssystem an einen Cluster weitergegeben werden. Die TSU nutzt dazu Informationen, die von der *Fault Detection Unit* (FDU), bezüglich der Leistungsfähigkeit des Clusters, bereitgestellt werden.

Die FDU nimmt eine überwachende Funktion ein. Zusammen bilden beide Einheiten mit den zughörigen Prozessorkernen die Cluster, welche die erste Hardware-Abstraktionsschicht des Systems darstellt. Alle Komponenten, inklusive des Betriebssystems, arbeiten nur auf Grundlage von virtuellen Prozessorknoten (den Clustern). Damit das Betriebssystem die Arbeitspakete dennoch intelligent auf die Cluster verteilen kann, stellt jede FDU Kennzahlen des eigenen Clusters bereit. Diese Kennzahlen beinhalten sowohl die freie Rechenkapazität, als auch einen Zuverlässigkeitswert des Clusters. Sind beispielsweise einige Prozessorkerne in einem Cluster defekt, senkt dies die Rechenkapazität.

Sowohl die TSU, als auch die FDU sind in Software implementiert und werden auf einem Prozessorkern des Clusters ausgeführt. Dies beseitigt die Gefahr eines *single point of failure* einer in Hardware festverdrahteten Verwaltungseinheit, da sich die Software von einem fehlerhaften zu einem funktionierenden Kern migrieren lässt.

2.3.2. Die Fehlererkennungseinheit FDU

Die Funktionsweise der FDU [Weis 2011] basiert auf einem Überwachungs- und Kontrollverfahren, welches als *MAPE-Zyklus*[12] [Kephart und Chess 2003] bezeichnet wird.

[12]Das Akronym MAPE steht für **M**onitoring, **A**nalyse, **P**laning und **E**xecution.

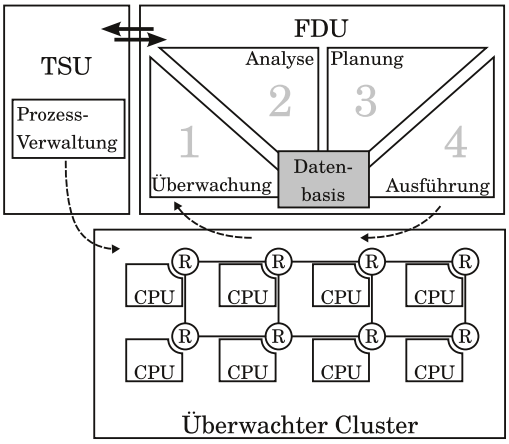


Abb. 2.8.: MAPE-Zyklus innerhalb der FDU. Die *Überwachung* liefert Vitalwerte der CPUs. Die *Ausführung* legt CPUs still oder reaktiviert sie. Die Datenbasis dient ebenfalls der TSU für die clusterinterne Prozessverwaltung.

Der MAPE-Zyklus ist eine Regelschleife und arbeitet innerhalb eines Clusters auf Basis von Zustandsinformationen der überwachten Prozessorkerne. Abbildung 2.8 zeigt eine schematische Darstellung der Arbeitsweise der FDU. Die einzelnen Arbeitsschritte der FDU teilen sich dabei in vier Phasen auf:

(i) Eine Überwachungsphase, bei der Vitalwerte der Prozessorkerne und des Netzes gesammelt werden.

(ii) Eine Analysephase, basierend auf den gesammelten Zustandsinformationen.

(iii) Eine Planungsphase, um beispielsweise drohende Temperaturprobleme zu beheben.

(iv) Eine Ausführungsphase, in der die geplanten Aktionen ausgeführt werden.

Die Überwachung der im Cluster enthaltenen Prozessorkerne realisiert die FDU mit Hilfe von Heartbeat-Nachrichten, die spezielle Zustandsinformationen der Prozessorkerne enthalten. Darunter sind Informationen wie die aktuelle Temperatur und die Anzahl korrigierter Fehler. Zusätzlich werden für jeden überwachten Prozessorkern ein Zeitpunkt und ein Intervall festgelegt, in denen der Prozessorkern eine Heartbeat-Nachricht absendet. Die FDU hält analog dazu eine Liste mit den erwarteten Ankunftszeiten der jeweiligen Heartbeat-Nachrichten, sodass eintreffende Nachrichten mit der aktuellen Zeit und dem erwarteten Zeitpunkt abgeglichen werden können. Wird eine Nachricht zu lange vermisst, sorgt eine ablaufende Wartefrist dafür, dass der Kern als defekt markiert wird.

Die Analysephase der FDU beruht auf der Auswertung der gesammelten Statusinformationen der einzelnen Prozessorkerne. Als Datenbasis dienen die prozessorinternen

Fehler- und Zustandsinformationen, wie sie beispielsweise mit der *Machine Check Architecture* (MCA) bei Intel's x86-Prozessoren bereitstellt werden.

Die Planungsphase arbeitet auf den Ergebnissen der Analyse und erzeugt Strategien, wie mit einem vorhandenen Problem/Fehler umgegangen wird. Als defekt markierte Prozessorkerne können beispielsweise abgeschaltet werden. Andersherum kann die FDU die Prozessorkerne ebenfalls wieder reaktivieren.

Die Ausführungsphase beinhaltet im Wesentlichen das Absenden von Nachrichten an die Prozessorkerne, die entweder abgeschaltet oder reaktiviert werden sollen. Alternativ sind auch weniger drastische Mittel einsetzbar. Sofern der Prozessor unterschiedliche Taktungen der Prozessorkerne unterstützt [Salihundam 2010; David 2011], sind mit Hilfe der Nachrichten auch Verringerungen/Erhöhungen der Taktfrequenzen initialisierbar.

Zusätzlich werden die Zustandsinformationen der FDU in der Ausführungsphase veröffentlicht. Dazu übermittelt die FDU die gesammelten Informationen sowohl an die TSU, als auch an das Betriebssystem. Beide Empfänger können dann auf Grundlage dieser Informationen Entscheidungen bezüglich der Prozessverwaltung fällen.

2.3.3. Das Heartbeat Konzept

Die Überwachungsphase der FDU setzt auf Heartbeat-Nachrichten und den darin enthaltenen Informationen auf. Innerhalb des TERAFLUX-Projektes wurden zwei Methoden der Informationsakquise, basierend auf den Heartbeat-Nachrichten, untersucht. Zum einen das *Poll*-Verfahren, bei dem die FDU aktiv mit einer Statusanfrage einen Prozessorkern auffordert, seinen internen Status (basierend auf der MCA) zu übermitteln. Alternativ dazu wurde ebenfalls das *Push*-Verfahren untersucht. Beim Push-Verfahren werden die Prozessorkerne beim Systemstart mit Hilfe von Konfigurationsnachrichten der FDU konfiguriert. In den Push-Nachrichten sind für jeden einzelnen Prozessorkern ein Startzeitpunkt und ein Intervall angegeben, die bestimmen, zu welchen Zeitpunkten der Prozessorkern seine Heartbeat-Nachricht absetzt.

2.3.3.1. Anforderungen der Heartbeat-Nachrichten

Da die Last innerhalb des Netzes nicht immer vollständig vorhersagbar ist, stellt das Konzept der Heartbeat-Nachrichten spezielle Anforderungen an das Kommunikationsnetz. So müssen die Statusnachrichten von den Anwendungsnachrichten isoliert werden, da andernfalls durch Blockierungen eine Berechnung der genauen Wartefristen innerhalb der FDU nicht möglich ist. Staus sind daher gerade bei hohen Belastungen des Netzes keine Seltenheit. Diese hätten bei einem gleichberechtigten Zugriff auf die Netzressourcen auch Einfluss auf die Latenz der Heartbeat-Nachrichten. Gleichzeitig muss das Netz

mit einer adaptiven Routing-Strategie ausgestattet sein, damit auf fehlerhafte Komponenten im Netz reagiert werden kann. Trotz des adaptiven Routings muss jedoch die Vorhersagbarkeit über die Ankunftszeiten gewahrt bleiben, was ein voll adaptives Routing ausschließt.

2.3.4. Das Kommunikationsnetz

Das Kommunikationsnetz wird im TERAFLUX-Projekt nur in Bezug auf Fehlertoleranz und Zuverlässigkeit behandelt. Daher wurde aus Vereinfachungsgründen das im TERAFLUX-Projekt angenommene Kommunikationsnetz als hierarchisch aufgebaute Topologie gewählt. Abweichend davon, wird in dieser Dissertationsarbeit das prozessorinterne Kommunikationsnetz als 2D-Gitter angenommen. Diese Art der Topologie skaliert besser mit der Anzahl der Prozessorkerne und durch die erhöhte Anzahl verfügbarer Verbindungsleitungen steht mehr Redundanz in Form von alternativen Pfaden zur Verfügung.

Das in dieser Dissertationsarbeit verwendete Router-Modell ist in Abbildung 2.9 dargestellt. Neben den zuvor beschriebenen Komponenten des generischen Routers wurde für diese Arbeit das Router-Modell erweitert.[13] Statt eines einzelnen Eingangspuffers, stehen hier zwei Pufferspeicher zu Verfügung. Die beiden Pufferspeicher einer Eingangsleitung sind jeweils einer Prioritätsklasse zugeordnet. Zusätzlich sind Demultiplexer vor den Eingangspuffern und Multiplexer nach den Eingangspuffern platziert. Die Aufgabe des Demultiplexers besteht darin, Nachrichten unterschiedlicher Prioritäten in die entsprechenden Puffer umzuleiten. Die Multiplexer sind zusätzlich mit der Arbiter-Einheit (BA) verbunden, die um ein sogenanntes *Quality of Service* (QoS) erweitert wurde. Das QoS entscheidet, ob und welche Nachrichten bevorzugt vom Router abgearbeitet werden sollen. Die hier verwendete QoS-Strategie geht bei der Auswahl der Nachrichten streng hierarchisch vor. Zuerst werden Eingangspuffer hoher Priorität auf Nachrichten geprüft, dann jene mit niedriger Priorität. Beinhaltet ein Eingangspuffer hoher Priorität eine Nachricht, wird diese immer bevorzugt bearbeitet. Eine mögliche Gefahr, dass dadurch Nachrichten niedrigerer Priorität *verhungern* (starvation problem), wird durch ein später näher beschriebenes TDMA-basierendes Sendemuster für Nachrichten hoher Priorität gelöst (siehe Abschnitt 3.1.1.2).

Innerhalb der Kreuzschiene (K) ist die globale Arbiter-Einheit (GA) integriert. Die GA beseitigt die Konflikte zwischen den konkurrierenden Zugriffen der Eingangspuffer auf gemeinsame Ausgangsleitungen. Auch die GA berücksichtigt die Prioritäten der Eingangspuffer bei der Kanalbildung.

[13] Bei dem Entwurf des Routers wurde speziell darauf geachtet, den Router mit möglichst wenig Hardware aufwand zu realisieren. Dies ist zwar immer eines der Hauptkriterien beim Entwurf des Netzes, jedoch wurde auch bewusst auf spezielle Hardware verzichtet, die die Leistungsfähigkeit des Netzes weiter steigern würde.

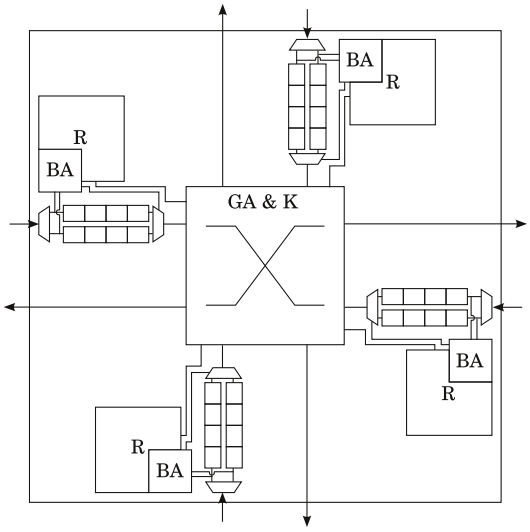


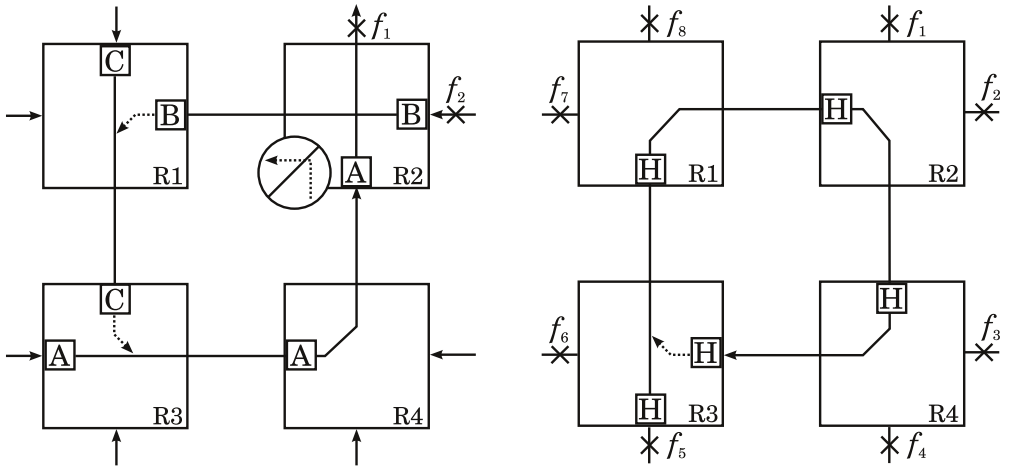
Abb. 2.9.: Konkretes Router Design in dieser Arbeit. R=Routerlogik, BA=Eingangspuffer Arbiter, GA=Globaler Arbiter, K=Kreuzschiene. Die Verbindung zur lokalen CPU wurde nicht eingezeichnet.

In der Abbildung 2.9 nicht dargestellt, sind die Funktionseinheiten der internen Fehlererkennung und der Anschluss des lokalen Prozessorkerns. Da die Fehlerinjektion der später folgenden Simulationen auf Grundlage von Verbindungsfehlern und Router-Fehlern basieren, und diese durch gezieltes Abschalten der entsprechenden Komponenten realisiert werden konnte, war eine Implementierung einer Fehlererkennungseinheit auf dem Router nicht notwendig.

Als Routing-Strategie kommt eine partiell adaptive Routing-Strategie zum Einsatz. Im fehlerfreien Fall wird das dimensions-orientierte XY-Routing für Anwendungsnachrichten verwendet. Heartbeat-Nachrichten werden mit der Routing-Strategie *Staircase-Routing* durch das Netz übertragen. Diese Routing-Strategie ist Teil der verbesserten Lastverteilung im Netz und wird im nächsten Kapitel genauer beschrieben und evaluiert.

Im Fehlerfall werden die Anwendungsnachrichten am betroffenen Router mit Hilfe des Turn-Models *West-First* [Glass und Ni 1992] verarbeitet. Für Heartbeat-Nachrichten dagegen wurde ein spezielles Protokoll implementiert [Garbade 2013]. Falls eine Heartbeat-Nachricht über eine fehlerhafte Komponente im Netz übertragen werden müsste, wird diese *nicht* auf der kürzesten Ausweichroute übertragen. Stattdessen wird diese einmalig über eine Verbindungsleitung übertragen, welche die Distanz zur FDU erhöht (*Artificial Deflection Protocoll, ADP*). Die künstliche Verzögerung der Heartbeat-Nachrichten ist notwendig für das Fehlerlokalisierungsverfahren in Kapitel 4.4.

Während mit West-First ein einzelner Fehler auf der Route einer Nachricht tolerierbar ist, erzeugen multiple Fehler eine erneute Deadlock-Gefahr. Im letzteren Fall lassen sich Situationen konstruieren, bei denen neue zyklische Abhängigkeiten entstehen. Abbildung



(a) Deadlock-Szenario durch multiple Fehler

(b) Einziges Deadlock-Szenario bei Heartbeat-Nachrichten

Abb. 2.10.: Deadlock Szenarios trotz adaptiver Routing-Strategien.

2.10(a) zeigt so einen konstruierten Fall. In dem Netz mit bidirektionalen Verbindungsleitungen, sind die mit f_1 und f_2 markierten Ausgangsleitungen[14] des abgebildeten Routers R2 defekt. Die Nachrichten A, B und C bilden durch die Fehler f_1 und f_2 eine gegenseitige zyklische Abhängigkeit. Grund für den Deadlock ist, dass durch beide Fehler die Nachricht A am Router R2 durch die verbotene Richtungsänderung *Nord-West* (siehe auch Abbildung 2.6(c)) blockiert wird[15].

Für die Heartbeat-Nachrichten gilt dies jedoch nicht. Ein Verfahren kombiniert aus virtuellen Kanälen und einem speziellen Sendeschema für Heartbeat-Nachrichten (siehe Abschnitt 3.1.2 & 3.1.3) sorgt dafür, dass Heartbeat-Nachrichten mit keinen anderen Nachrichten kollidieren können. Die Kollisionsfreiheit der Heartbeat-Nachrichten stellt sicher, dass es genau ein mögliches Szenario gibt, bei dem tatsächlich ein Deadlock entstehen kann. Die Heartbeat-Nachricht H in Abbildung 2.10(b) wird durch die acht fehlerhaften Ausgangsleitungen der Router R1 - R4 gewissermaßen eingefangen. Sofern die Paketlänge der Heartbeat-Nachricht lang genug ist, entsteht dabei der Deadlock an R3. Ist die Paketlänge kürzer, besteht die Gefahr eines Livelocks, bei dem, ohne entsprechende Maßnahmen, die Heartbeat-Nachricht im Kreis herum übertragen wird.

Darüber hinaus sind fehlertolerante Routing-Strategien kein Bestandteil dieser Dissertationsarbeit und führten am Kern der Arbeit vorbei. Auch hier sei auf die Fachliteratur [Duato 2002; Dally und Towles 2003; Flich und Bertozzi 2010] für weitere Informationen zu fehlertoleranten Routing-Strategien verwiesen.

[14] In diesem Fall kann der Router weiterhin aus den entsprechenden Richtungen Nachrichten empfangen, aber keine absetzen.

[15] Auch die Richtungsänderung Nord-Süd (180° Wende) würde erneut neue Abhängigkeitsverhältnisse schaffen, wodurch sich weitere Deadlocks konstruieren ließen.

2.4. Fazit

Da die bisherige TERAFLUX-Architektur einen gleichberechtigen Zugang zum Netz vorsieht, kann keine Vorhersagbarkeit bezüglich der Ankunftszeiten für Heartbeat-Nachrichten hergestellt werden. Um die Wartefristen exakt bestimmen zu können, musste die TERAFLUX-Architektur erweitert werden, um die folgenden Eigenschaften zu erfüllen:

(i) Isolation der Heartbeat-Nachrichten von Anwendungsnachrichten.

(ii) Isolation der Heartbeat-Nachrichten untereinander.

Die Implementierung und Auswertung dieser Architektur-Erweiterungen sind der erste Teil dieser Dissertationsarbeit und werden in Kapitel 3 im Detail beschrieben und Evaluiert.

Der durch die Erweiterungen erzeugte Determinismus der Heartbeat-Nachrichten, ist anschließend das Fundament für das 4. Kapitel. Dort wird das Fehlerlokalisierungsverfahren beschrieben. Teil der Beschreibung sind weitere Anpassungen, die dann jedoch speziell den Heartbeat-Mechanismus betreffen. Zusätzlich findet in Kapitel 4 eine Diskussion bezüglich multipler Fehler und toroidaler Netztopologien statt.

3. Einfluss der Heartbeat-Nachrichten auf Anwendungsnachrichten

Dieses Kapitel beschreibt die technische Umsetzung der Anforderungen der FDU-basierenden Fehlertoleranz an das Kommunikationsnetz und geht weiter auf deren Einflüsse auf das Kommunikationsnetz ein. Gleichzeitig dient das Kapitel als Machbarkeitsstudie und wird mögliche Engpässe, erzeugt durch die Heartbeat-Nachrichten, identifizieren. Zusätzlich wird mit Hilfe einer neuen Routing-Strategie gezeigt, wie die Wirkung der Engpässe auf Anwendungsnachrichten entspannt werden kann.

3.1. Anforderungsimplementierung

Aus dem Abschnitt 2.3.2 ist bekannt, dass das FDU-basierende Fehlertoleranzverfahren spezielle Anforderungen an die Prozessorarchitektur stellt. Im folgenden Abschnitt wird die technische Umsetzung dieser Anforderungen aus Sicht des Kommunikationsnetzes beschrieben.

Neben den gehärteten kerninternen Systemen wie dem Bussystem und dem lokalen Speicher, fordert die FDU weitere Eigenschaften, die den Transport der Heartbeat-Nachrichten im Netz betrifft. Während der Prozessor für das korrekte Abschicken der Nachricht verantwortlich ist, liegt die Nachrichtenübertragung über das Kommunikationsnetz außerhalb seiner Kontrolle. Deshalb muss das Netz die nötigen Eigenschaften zur Verfügung stellen, um damit Heartbeat-Nachrichten übertragen zu können, ohne dass diese dabei durch *äußere* Einflüsse verzögert werden. Daher liegt der Schwerpunkt der Implementierung darin, Heartbeat-Nachrichten kollisionsfrei durch das Netz zu transportieren. Da die Heartbeat-Nachrichten sowohl durch Anwendungsnachrichten, als auch durch andere Heartbeat-Nachrichten blockiert werden können[1], ergeben sich folgende Anforderungen: Anwendungsnachrichten und Heartbeat-Nachrichten müssen gegenseitig isoliert im Netz verarbeitet werden. Zusätzlich müssen Heartbeat-Nachrichten untereinander isoliert werden, um gegenseitige Einflüsse zu vermeiden.

[1] Beispielsweise durch den konkurrierenden Zugriff auf eine gemeinsame Ausgangsleitung eines Routers.

3.1.1. Verfahren zum Isolieren von Nachrichten

Die möglichen Verfahren, Nachrichten voneinander zu isolieren, variieren zwischen der räumlichen und zeitlichen Trennung. Abbildung 3.1 veranschaulicht schematisch, welche Optionen für das Separieren verschiedener Nachrichtentypen zur Verfügung stehen. Dabei teilt sich die räumliche Trennung zusätzlich in physisch und logisch auf.

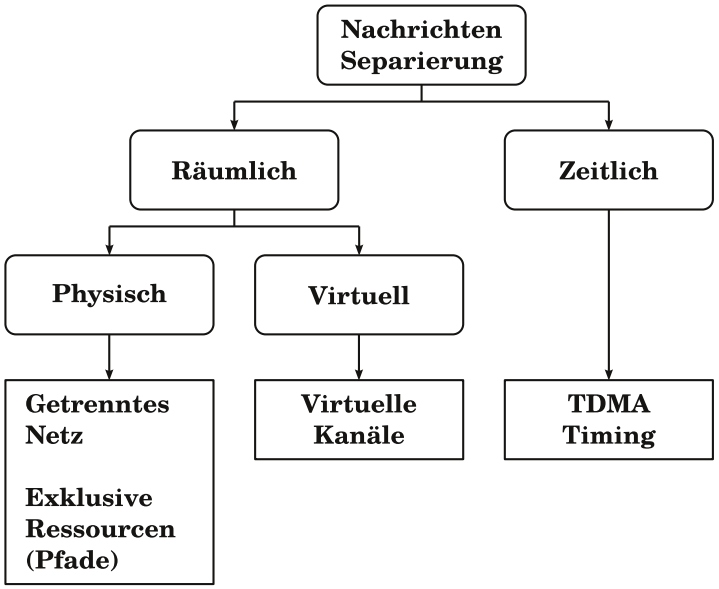


Abb. 3.1.: Mögliche Methoden um Nachrichten voneinander zu trennen.

3.1.1.1. Physische vs. logische Isolation

Die physische Trennung erfolgt durch physikalisch getrennte Netze. Ein prominentes Prozessorbeispiel eines solchen Netzaufbaus ist der Tile64 von Tilera [Bell 2008]. Der Tile64 verfügt über fünf physisch getrennte Netze von denen vier paketorientierte Verbindungen unterstützen (*Packet Based Routing*). Das fünfte Netz ist verbindungsorientiert konfiguriert (*Curciut-Switched Routing*). Die Datenströme der vier paketorientierten Netze teilen sich in Interprozesskommunikation, Ein-Ausgabe-Geräte und Speicherzugriffe auf. Anfragen des Cache-Transfers, bei dem Daten direkt zwischen lokalen Caches ausgetauscht werden, verwenden das vierte Netz und sind damit von anderen Speicherzugriffen (beispielsweise direkt auf den Hauptspeicher) isoliert. Der offensichtliche Vorteil dieses Aufbaus resultiert aus der geringeren Konkurrenz innerhalb der Netze und den damit verbundenen geringen Nachrichtenlatenzen [Wentzlaff 2007]. Dem gegenüber stehen die Implementierungskosten für jedes dieser unabhängigen paketorientierten Netze. Jeder Router benötigt jeweils eine vollständig unabhängige Pipeline für jedes einzelne Netz. Das bedeutet, dass die geringe Latenz der Nachrichten durch einen höheren Hardware-Aufwand des Gesamtnetzes erkauft wird.

Die alternative Methode um Datenströme räumlich voneinander zu isolieren, stellen die logischen Netze dar. Hier wird ein einzelnes physisches Netz genutzt um mehrere logische Netze anzulegen[2]. Die Isolation der Datenströme wird in dieser Variante durch Verarbeitungsregeln erreicht, welche auf einzelne Nachrichtentypen angewendet werden. Am Beispiel des Tile64 würde dies bedeuten, dass die fünf verschiedenen Datenströme in Klassen untergliedert und Regeln aufstellt werden, wie die einzelnen Klassen zu behandeln sind. In Kommunikationsnetzen werden diese Regeln auch als *Quality of Service (QoS)* bezeichnet und regeln dort sowohl die zugesicherte Bandbreite als auch einen Priorisierungsmechanismus, um speziellen Klassen geringe Latenzen durch bevorzugte Verarbeitung zu ermöglichen. Zu den Mehrkosten eines solchen Netzentwurfs tragen zum Einen die notwendigen Erweiterungen der Arbitrierungseinheit bei, zum Anderen müssen Pufferspeicher hinzugefügt werden, um blockierte Nachrichten zwischenspeichern zu können, bis diese gemäß dem QoS verarbeitet werden. Ein weiterer Effekt ergibt sich durch aus der Priorisierung der Nachrichten. Nachrichten mit niedriger Priorität können in logischen Netzen mitunter sehr lange Wartezeiten erreichen, wenn diese durch Nachrichten hoher Priorität blockiert werden.

3.1.1.2. Temporale Isolation

Bei der temporalen Aufteilung des Netzes handelt es sich um eine verwandte Methode, wie sie bereits für die logischen Netze beschrieben wurde, da auch bei diesem Verfahren nur ein physisches Netz zur Verfügung steht. Jedem Sender werden spezielle Zeitschlitze zugewiesen, in denen der Sender das Recht erhält, eine Nachricht abzusetzen. Im Bereich der Kommunikationsnetze wird dieses Schema als *Time Division Multiple Access (TDMA)* bezeichnet und häufig zur Zugriffsregulierung bei mehreren gleichberechtigten Kommunikationsteilnehmern verwendet. Vereinfacht ausgedrückt und in der einfachsten Form eines TDMA-Schemas, wird ein fester Zeitraum (eine Runde) in gleichlange Zeitschlitze zerlegt und jedem Sender ein Schlitz pro Runde zugeordnet. Das jeweilige Senderecht liegt dann bei dem Sender, in dessen Zeitschlitz sich das System gerade befindet. Eine Runde startet erneut, wenn alle Zeitschlitze einmal durch iteriert wurden und jeder Sender das Recht zu senden hatte. Wurden die Parameter der Zeitschlitze korrekt gewählt, kann auf diese Weise garantiert werden, dass Pakete nicht miteinander kollidieren. Ein weiterer Vorteil eines TDMA-Schemas liegt in der Wartbarkeit und am geringen Aufwand den Zugriff zu regulieren. Beides lässt sich von einer (de)zentralen Stelle im System verwalten. Zudem kann mit Hilfe der Vorhersagbarkeit, welcher Sender zu welchem Zeitpunkt etwas senden darf, auf die Adressierungen des Senders verzichtet werden [Goossens 2003], da der ursprüngliche Sender einer Nachricht anhand des Zeitschlitzes in dem die Nachricht abgesetzt wurde, genau bestimmt werden kann.

Zwar ist die statische Vergabe der Zeitschlitze und dessen Länge in einfachen Bussystemen praktikabel, jedoch besteht dabei auch die Chance einen Großteil der Ressourcen

[2] Auch virtuelles Netz genannt.

nur sporadisch zu nutzen. Denn nicht jeder Sender nutzt seinen zugewiesenen Zeitschlitz auch tatsächlich zur Kommunikation. Hinzu kommt, dass in dieser Arbeit statt einem Bussystem eine 2D-Gitter Topologie zum Einsatz kommt, bei der die reine TDMA basierende Zugriffskontrolle wegen der hohen Zahl an redundanten Verbindungen nur bedingt effizient ist.

3.1.2. Logische Netze trennen Heartbeats und Applikationsnachrichten

Um Heartbeat-Nachrichten einerseits von Anwendungsnachrichten zu isolieren und andererseits daran zu hindern, sich untereinander zu blockieren, wurde für die Implementierung der Nachrichtenisolation eine Kombination aus logischen Netzen und der temporalen Isolation gewählt.

Durch die logischen Netze werden die Heartbeat-Nachrichten und die Anwendungsnachrichten voneinander isoliert. Ferner wurde den Heartbeat-Nachrichten per Arbitrierungsregel eine höhere Priorität im logischen Netz zugeordnet. Diese Priorität ist strikt und sorgt dafür, dass eine eingehende Heartbeat-Nachricht immer bevorzugt weitergeleitet wird. Zusätzlich wurde der Router um eine Multiplexfunktion erweitert. Die Übertragung von Anwendungsnachrichten, die sich bereits im *Transit* befinden, können damit ad hoc gestoppt werden, um dann Heartbeat-Nachrichten zwischendurch versenden zu können. Ist der Ausgang der Router wieder freigegeben, wird die alte Verbindung wieder hergestellt und der Rest der Anwendungsnachricht wird übermittelt. Diese Funktionalität verhindert, dass ein bereits reservierter oder benutzter Ausgang eines Routers eine Heartbeat-Nachricht blockiert und somit verzögert. Im Abschnitt A des Anhangs dieser Arbeit veranschaulichen zwei Beispiele die Funktionsweise des Multiplexers und illustrieren die Auswirkung dieses Multiplexverfahrens auf Heartbeat-Nachrichten und Anwendungsnachrichten.

3.1.3. TDMA für Heartbeat-Nachrichten

Um gegenseitiges Blockieren der Heartbeat-Nachrichten zu vermeiden, wird ein TDMA-Schema angewendet, das den Zugriff der Heartbeat-Nachrichten auf das Netz limitiert. Abbildung 3.2 verdeutlicht das Gesamtkonzept des TDMA-Schemas anhand eines 5×5 großen Clusters. Auf der linken Seite der Abbildung ist das Netz schematisch aufgetragen. Die unterschiedlichen Grautöne der Knoten symbolisieren die Distanz zur FDU. Analog dazu ist, auf der rechten Seite der Abbildung, ein Zeitstrahl zu sehen. Dieser illustriert die Zeitpunkte in denen die Heartbeat-Nachrichten abgesendet und empfangen werden. Auch hier entspricht die Tönung der Distanz.

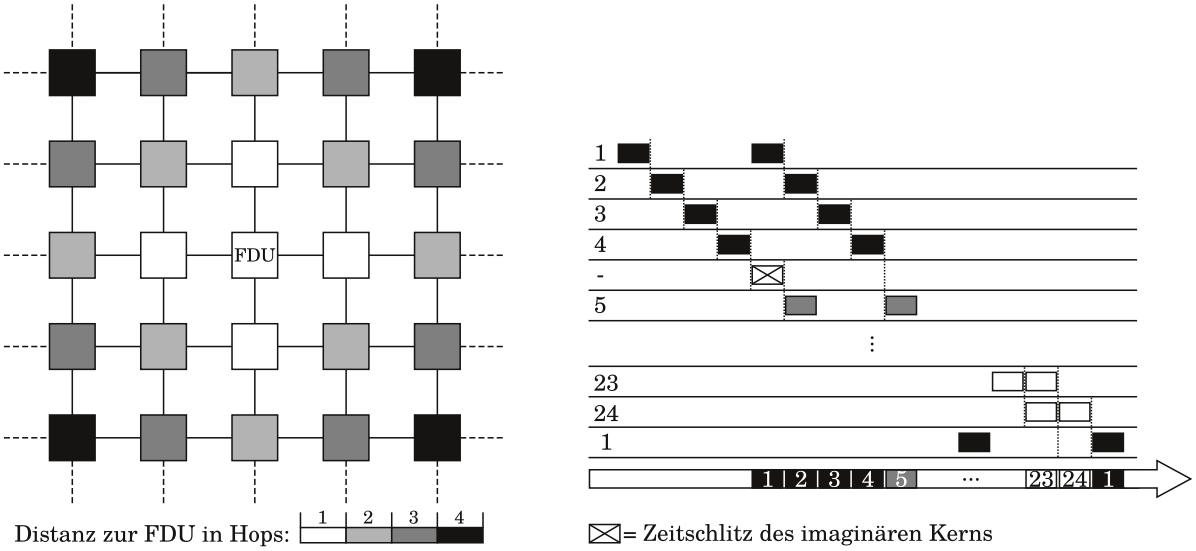


Abb. 3.2.: Schematische Darstellung des TDMA-Schemas eines 5×5 Clusters.

Wie zu erkennen ist, senden zunächst alle Kerne der Distanz 4 ihre jeweiligen Heartbeat-Nachrichten ab. Die gefärbten Rechtecke illustrieren den Zeitschlitz, in dem die Nachrichten vom jeweiligen Kern abgesetzt wurden.

Um das TDMA-Schema für die Heartbeat-Nachrichten zu erstellen sind drei Schritte nötig:

(i) Festlegen der Reihenfolge in der die Prozessorkerne senden.

(ii) Berechnung der Dauer der Zeitschlitze.

(iii) Einfügen spezieller leerer Zeitschlitze um Überlagerungen aufzulösen.

Der erste Schritt legt die Reihenfolge fest, in der die Prozessorkerne senden dürfen. Im zweiten Schritt wird die Länge des Zeitschlitzes für die Sendeberechtigung ermittelt.

Die Länge der Zeitschlitze ist für alle Prozessorkerne gleich, da die Nachrichtenlänge aller Heartbeat-Nachrichten aus Vereinfachungsgründen gleich ist. Der letzte Schritt fügt spezielle leere Zeitschlitze ein, die eine Überlagerung verschiedener Heartbeat-Nachrichten vermeiden. Die leeren Zeitschlitze kompensieren dabei einen von Fehlern im Netz unabhängigen Überlagerungseffekt, der auftreten kann, wenn zwei Prozessorkerne mit unterschiedlicher Distanz eine Heartbeat-Nachricht zur FDU absenden.

Die Reihenfolge der Sendeberechtigung basiert auf der Distanz d zwischen der FDU und dem sendendem Prozessorkern. Absender mit gleicher Distanz werden zu Gruppen K_d zusammengefasst. Da die zugrundeliegende Netztopologie eines 2D-Gitters entspricht und die Adressierung der einzelnen Prozessorkerne mit den Koordinaten eines Punktes im kartesisches Koordinatensystem identisch ist, kann die Distanz zwischen einem

Prozessorkern i und der FDU f über die Manhattan Distanz Metrik berechnet werden:

$$d_i = md(x, y) = |x_i - x_f| + |y_i - y_f|$$

Die Reihenfolge in der die Gruppen abgearbeitet werden basiert auf d beginnend mit der Gruppe der größten Distanz[3]. Innerhalb einer Gruppe ist die Sendereihenfolge irrelevant, da alle Prozessorkerne einer Gruppe die gleiche Distanz zur FDU haben und damit die gleiche Zeit benötigen, um die Heartbeat-Nachricht zu übermitteln. Dies gilt solange sichergestellt ist, dass die Heartbeat-Nachrichten nacheinander und niemals gleichzeitig abgesetzt werden.

Nachdem die Reihenfolge der Sender festgelegt wurde, wird die Länge der Zeitschlitze ermittelt. Anders als bei dem oben beschriebenen vereinfachtem TDMA-Schema besteht der hier eingesetzte Zeitschlitz aus einer Sendephase P_s und einer Abkühlphase P_c. Für die Dauer der Sendephase ist es dem entsprechenden Prozessorkern möglich die Heartbeat-Nachricht abzusetzen. Die Länge P_s wird dabei durch die Länge der Heartbeat-Nachricht bestimmt. Die Abkühlphase folgt direkt der Sendephase und ist ebenfalls Teil des Zeitschlitzes eines Prozessorkerns. Da sich die Zeitschlitze nicht überlappen, wirkt die Abkühlphase P_c wie ein zusätzlicher Puffer zwischen den Sendezeitpunkten der Heartbeat-Nachrichten. Die gewählte Länge von P_c entspricht dabei dem zeitlichen Abstand zweier aufeinanderfolgenden Heartbeat-Nachrichten, die bei der FDU eintreffen. Somit ist $P_c = 0$ ein pausenloser Strom aus Heartbeat-Nachrichten, welcher bei der FDU eingeht.

Im dritten Schritt werden leere Zeitschlitze P_p zwischen den jeweiligen Distanz-Gruppen des TDMA-Schema eingefügt, um die unterschiedlichen Laufzeiten der Heartbeat-Nachrichten zu kompensieren. Ohne diesen zusätzlichen Puffer würde sich die letzte Heartbeat-Nachricht der Gruppe K_d mit der ersten Nachricht der folgenden Gruppe K_{d-1} überlagern. Die Folge wäre, dass spätestens im Router der FDU ein Konflikt dieser beiden Heartbeat-Nachrichten auftritt. Die Länge von P_p entspricht einer vollständigen Sende- und Abkühlphase:

$$P_p = \begin{cases} P_s + P_c, & \text{wenn } d_i > d_{i+1} \text{ (Gruppenübergang)} \\ 0, & \text{sonst} \end{cases}$$

Nach jeder vollständigen TDMA-Runde beginnt das TDMA-Schema von vorn. Der Übergang zwischen den Runden lässt sich optimieren, in dem der Start der anschließenden Runde bereits beginnt, während die alte Runde noch nicht ganz abgeschlossen ist. Die Anzahl der überlappenden Zeitschlitze ist dabei gleich der maximalen Distanz aller Prozessorkerne zur FDU:

$$d_{max} = max(d_i), \forall i \in K \tag{3.1}$$

[3]Dieses Verfahren ist nicht auf diese spezielle Reihenfolge beschränkt. Denkbar sind auch umgekehrte Reihenfolgen, bei denen sich lediglich die Berechnung spezieller Nachrichtenüberlagerungen ändert.

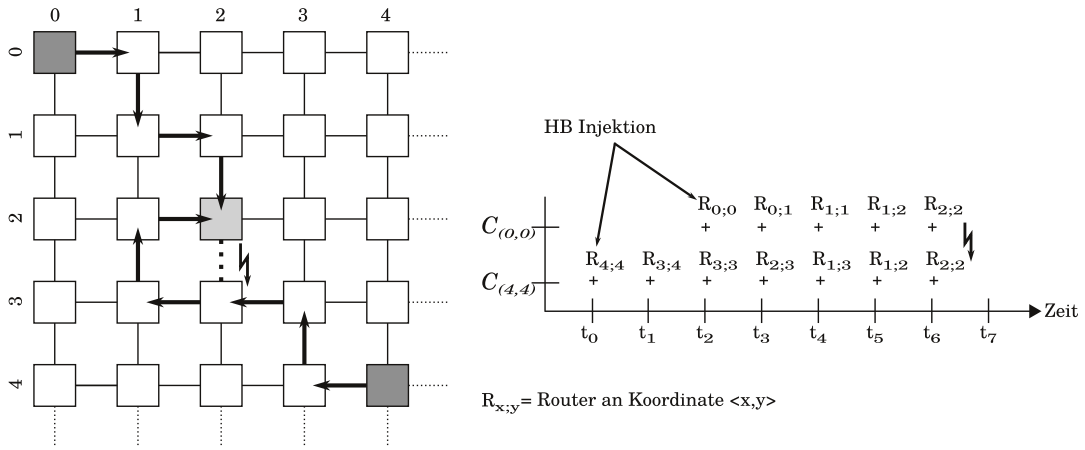


Abb. 3.3.: Kollision zweier Heartbeat-Nachrichten durch einen Fehler in einer Verbindungsleitung

Fasst man nun die einzelnen Schritte mit der Optimierung zum Rundenneustart zusammen, ergibt sich mit

$$Z = |K| \times (P_s + P_c + P_p - O), O = \begin{cases} d_{max}, & \text{wenn } d_i < d_{i+1} \\ 0, & \text{sonst} \end{cases} \tag{3.2}$$

die Berechnung der Länge einer gesamten TDMA-Runde im Falle des fehlerfreien Netzes.

Fehler bedeuten für den Lokalisierungsmechanismus mögliche Gefahren. In Abbildung 3.3 wird ein Beispielszenario skizziert, in dem bereits der erste Fehler im Netz zu einem generellen Problem führt. Das Beispiel illustriert eine Kollision zwischen zwei Heartbeat-Nachrichten. Diese Kollisionen, die vormals mit dem TDMA-Schema verhindert wurden (vgl. Abschnitt 3.1.1.2), treten auf, wenn sich eine Heartbeat-Nachricht durch einen Fehler im Netz nicht wie geplant an die vorgeschriebene Route zur FDU halten kann. Im gegebenen Beispiel wird die Heartbeat-Nachricht an der Verbindungsleitung $R_{(2,3)} \overset{N}{\rightarrow} R_{(2,2)}$ blockiert und gezwungen eine alternative Route um die fehlerhafte Verbindungsleitung zu verwenden. Dies führt dazu, dass der Router $R_{(2,2)}$ zum Zeitpunkt t_6 zeitgleich die Heartbeat-Nachrichten von $C_{(0,0)}$ und $C_{(4,4)}$ entgegen nimmt und entscheiden muss, welche der beiden Nachrichten als nächstes weitergeleitet wird. Da $R_{(2,2)}$ keine Informationen über die eigentliche Reihenfolge der Heartbeat-Nachrichten hat, wird nach dem Round-Robin Muster entschieden, welche Nachricht bevorzugt weitergeleitet wird. Das wiederum entspricht der Verletzung der Isolation von Heartbeat-Nachrichten, woraus ein nicht deterministisches Verhalten der Heartbeat-Nachrichten resultiert.

Gelöst wird dieses generelle Problem durch eine dynamische Anpassung des TDMA-Schemas. Die Kernidee bei der Anpassung besteht darin, den zeitlichen Abstand zwischen jeweils zwei Heartbeat-Nachrichten in Form eines zusätzlichen (zeitlichen) Puffers zu vergrößern. In dem hier skizzierten Beispiel würde der Einsatz eines Puffers mit minimaler Größe bedeuten, dass der Sendezeitpunkt des Kerns $C_{(0,0)}$ von t_2 auf t_3 verschoben

und damit die Kollision verhindert wird. Hierzu verfügt der Lokalisierungsmechanismus bereits über die notwendige Anpassungs-Routine. Es handelt sich dabei um die Rekonfiguration des Sendemusters im Falle eines lokalisierten Fehlers (vgl. Abschnitt 4.3). Da hier die Sendezeitpunkte angepasst werden, wird dieser Mechanismus eingesetzt, um die Mindestabstände zwischen den jeweiligen Heartbeat-Nachrichten dynamisch zu kontrollieren. Der zeitliche Puffer innerhalb des TDMA-Schemas wird für alle im Schema enthaltenden Prozessorkerne angewendet. Dazu wird die Gleichung 3.2 der Rundenlänge um die Puffergröße B für jeden Prozessorkern erweitert:

$$Z = |K| \times (P_s + (P_c + B) + P_p - O)$$

wobei P_c (Cooldown-Phase) um die zusätzliche Wartezeit B erweitert wurde. Dies führt zu zwei Effekten:

(i) Die Abstände zwischen zwei Heartbeat-Nachrichten vergrößern sich

(ii) Durch die vergrößerten Abstände, vergrößert sich ebenfalls die Dauer der TDMA-Runde.

Bei der Wahl der Puffergröße spielt vor allem die Fehlerwahrscheinlichkeit eine wesentliche Rolle. Ist die Fehlerwahrscheinlichkeit gering und multiple Fehler werden sukzessive erwartet, kann der Puffer auf die minimale Größe von

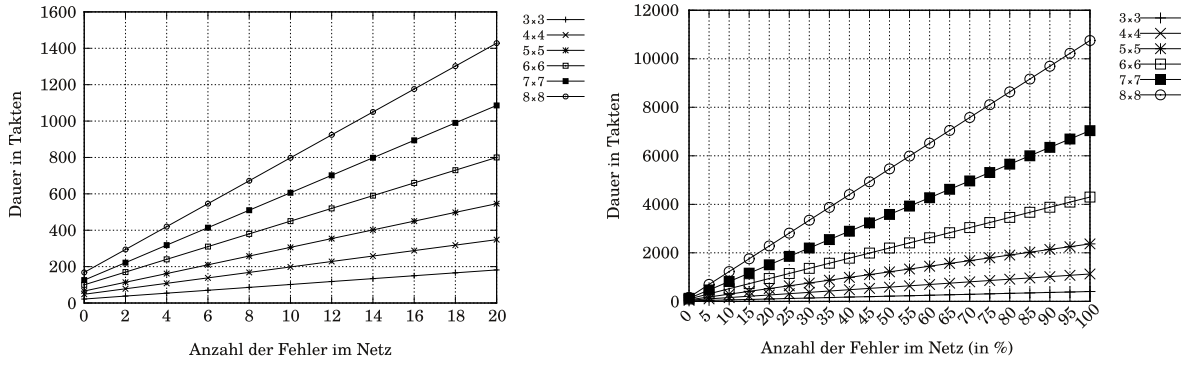
$$B = H \times P_{length}$$

festgelegt werden. Wobei H für die Anzahl künstlich hinzugefügten Hops und P_{lenth} für die Latenz der Router-Pipeline steht.

Wird das Push-Verfahren für die Heartbeat-Übermittlung angewendet, wird jeder überwachte Prozessorkern mit diesem Intervall von der zuständigen FDU vorkonfiguriert und sendet dann gemäß dem TDMA-Schema die Heartbeat-Nachrichten. Beim Poll-Verfahren richtet sich die FDU selbst nach dem TDMA-Schema, um die Poll-Nachrichten an die Kerne zu senden.

Geht man hingegen von einer hohen Fehlerwahrscheinlichkeit aus und werden zusätzlich simultan auftretende multiple Fehler in Betracht gezogen, muss bereits im fehlerfreien Fall ein hinreichend großer Puffer bei der Erstellung des TDMA-Schemas eingeplant werden. In den Abbildungen 3.4(a) und 3.4(b) wurde anhand zweier Beispiele die Auswirkung der Puffergrößen auf die Rundenlängen dargestellt. Neben der Standardrundenlänge ohne Fehler ($x = 0$) sind verschieden viele tolerierbare Fehler im Netz angenommen. Wobei Abbildung 3.4(a) ein Beispiel mit absoluter Fehlerzahl von $0, \ldots, 10$ ist und Abbildung 3.4(b) die Anzahl der zu tolerierenden Fehler prozentual zur Anzahl der verfügbaren Verbindungsleitungen innerhalb des Clusters berücksichtigt.

Wie zu erkennen ist, sind von den steigenden Rundenlängen besonders die großen Cluster-Größen betroffen, jedoch steigen die Rundenlängen insgesamt linear, sodass eine per



(a) Fehlerzahl absolut (b) Fehlerzahl relativ

Abb. 3.4.: Auswirkungen auf das TMDA-Schema durch die zusätzlich einfügten Puffer zur Tolerierung multipler Fehler.

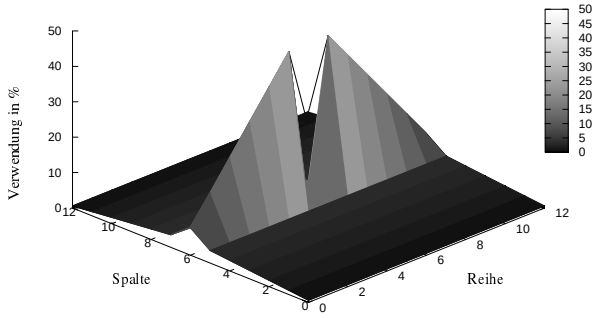
Design vorgegebene Anzahl an tolerierbaren Fehlern leicht in das TDMA-Schema eingerechnet werden kann. Ist beispielsweise eine maximale Rundenlänge von $10\mu s$ in einem 8×8 Cluster vorgesehen und das Netz mit 500MHz getaktet, lässt sich damit theoretisch eine tolerierbare Ausfallrate von $\approx 45\%$ erreichen, ohne dabei die Isolation der Heartbeat-Nachrichten durch Fehler im Netz zu verlieren[4].

Es ist also leicht möglich die Grundvoraussetzungen für einen korrekten Betrieb der Fehlerlokalisierung auch bei multiplen Fehlern zu gewährleisten. Offen ist jedoch die Frage, in wieweit sich Entstehungszeitpunkte (zeitliches Muster) multipler Fehler und deren jeweiliger Ort (räumliche Muster) auf die Genauigkeit des Lokalisierungsverfahrens auswirken. Der Abschnitt 4.6 des nächsten Kapitels widmet sich dieser Fragestellung und zeigt mögliche Kombinationen zwischen den Typen beider Muster, bei denen keine eindeutige Lokalisierung möglich ist.

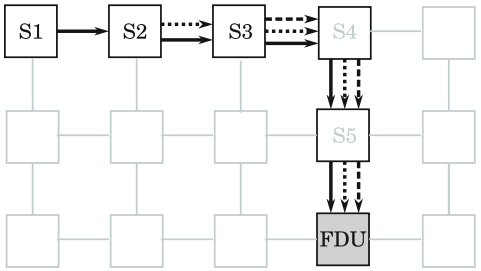
3.2. Engpässe

Es gibt nun zwei Mechanismen, um die einzelnen Nachrichtentypen voneinander zu isolieren. Auf der einen Seite, gibt es die logischen Netze, bei denen Heartbeat-Nachrichten von Anwendungsnachrichten isoliert sind. Auf der anderen Seite das TDMA-Schema, welches die Isolation der Heartbeat-Nachrichten untereinander realisiert. Da beide Mechanismen unabhängig voneinander sind, können beide miteinander kombiniert werden. Die Kombination bleibt aus Sicht der Anwendungsnachrichten jedoch nicht ohne Folgen. Die prioritätenbasierende Flusskontrolle und der konstante Strom der Heartbeat-Nachrichten durch das TDMA-Schema, üben einen direkten Einfluss auf Anwendungsnachrichten aus.

[4]Es ist an dieser Stelle darauf hingewiesen, dass eine Ausfallrate dieser Größenordnung im Kommunikationsnetz bereits einen deutlichen Einfluss auf den Prozessor hat und dass in der Regel deutlich früher fehlerhafte Komponenten ausgetauscht werden [Schroeder und Gibson 2010; Nightingale 2011].



(a) Belastung des Netzes durch Heartbeat-Nachrichten.



(b) $S1, S2$ und $S3$ blockieren Anwendungsnachrichten von $S4$ zu $S5$.

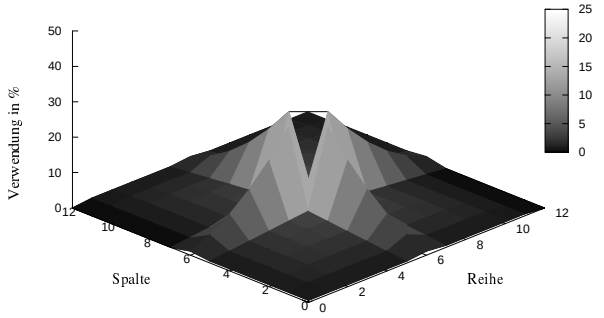
Abb. 3.5.: Lastverteilung durch Heartbeat-Nachrichten unter der Verwendung der XY-Strategie.

So ist es wahrscheinlich, dass die Anwendungsnachrichten auf dem Weg vom Sender zum Empfänger einen konkurrierenden Zugriff mit einer Heartbeat-Nachricht zu erwarten haben. Aus Sicht der FDU-basierende Fehlertoleranz ist ein möglichst dichter Strom aus Heartbeat-Nachrichten wünschenswert, da auf diese Weise alle Prozessorkerne in möglichst kurzen Abständen ihren internen Gesundheitszustand übertragen können. Probleme eines oder mehrerer Prozessorkerne können damit früher identifiziert werden. Ein konstanter Strom aus eng gestaffelten Heartbeat-Nachrichten erhöht jedoch die Wahrscheinlichkeit konkurrierender Zugriffe im Netz. Dies ist besonders dort zu erwarten, wo alle Heartbeat-Nachrichten zusammenlaufen; am Router der FDU. Um den Einfluss der Heartbeat-Nachrichten einschätzen zu können werden im Folgenden mögliche Engpässe und deren Einflussfaktoren identifiziert.

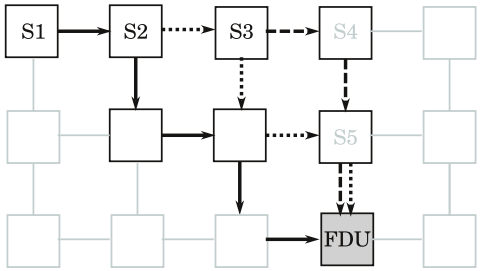
Wie eingangs beschrieben, befindet sich die FDU im Idealfall im Zentrum einer zu überwachenden Prozessorgruppe. Auf diese Weise sind die Abstände der Prozessorkerne gleichmäßig verteilt und vereinfacht die Koordination der Kommunikation mit der FDU. Dies bedeutet aber auch, dass alle Heartbeat Nachrichten genau an diesem zentralen Punkt zusammenlaufen, was als *Narrow-Cast* [Flich und Bertozzi 2010] bezeichnet wird. In Abbildung 3.5(a) stellt ein Histogramm die durchschnittliche Belastung der Router durch Heartbeat-Nachrichten dar. In dem hier gewählten Beispiel wurde das Sendemuster so engmaschig konfiguriert, dass in jedem Netztakt eine Heartbeat-Nachricht bei der FDU eintrifft und somit eine maximale Dichte von Heartbeat-Nachrichten im Netz vorherrscht. Die Achsen benannt als *Reihe* und *Spalte* definieren den Ort der Router im 2D-Gitter. Auf der z-Achse ist die prozentuale Belastung durch Heartbeat-Nachrichten aufgetragen.

Deutlich zu erkennen ist die stetig steigende Belastung der Router in der Spalte 6, je mehr man sich dem Router der FDU im Zentrum nähert[5]. Die direkt benachbarten Rou-

[5] Aus Gründen der Übersicht, wurde in den Abbildungen 3.5(a) und in Abbildung 3.6(a) der Wert des zentralen Routers weg gelassen.



(a) Belastung des Netzes durch Heartbeat-Nachrichten.



(b) $S1$ und $S2$ verwenden alternative Routen, dadurch entspannt sich die Lage zwischen $S4$ und $S5$.

Abb. 3.6.: Lastverteilung durch Heartbeat-Nachrichten unter der Verwendung des Staircase-Strategie.

ter der FDU in Spalte 6 erfahren in diesem Beispiel jeweils eine fast 50%ige Belastung durch Heartbeat-Nachrichten. Es muss also erwartet werden, dass es in dieser Spalte vermehrt zu Verzögerungen von Anwendungsnachrichten kommt, wenn sie diesen Bereich des Netzes passieren müssen. Verdeutlicht wird dies noch einmal in Abbildung 3.5(b). Die Heartbeat-Nachrichten von den Kernen $S1, S2$ und $S3$ (Pfade als Pfeile markiert) folgen der XY-Strategie. Ist das TDMA-Schema sehr engmaschig konfiguriert besteht die Möglichkeit, dass Anwendungsnachrichten von $S4$ Richtung Süden solang bei $S4$ blockiert werden, bis alle Heartbeat-Nachrichten diesen Bereich passiert haben. Dies ist besonders dann problematisch, wenn die Puffer eines Routers in einem solchen Brennpunkt vollständig gefüllt sind und ein Rückstau von Anwendungsnachrichten entsteht, der sich auch in andere Bereiche des Netzes ausbreitet.

Zwar lässt es sich in diesem Netz nicht verhindern, dass Anwendungsnachrichten von Heartbeat-Nachrichten blockiert werden, jedoch ist es möglich mit einer angepassten Routing-Strategie die maximale Wartezeit durch Blockaden zu verringern. In Abbildung 3.6 wird die Staircase-Strategie anhand des gleichen Netz- und TDMA-Schemas illustriert. Anhand von Abbildung 3.6(b) wird deutlich, dass die Last durch Heartbeat-Nachrichten im Vergleich zur XY-Strategie durch die Verwendung von mehr Netzressourcen besser verteilt wird. Die Blockaden, die dennoch entstehen, verteilen sich im Netz und wirken somit zwar auf mehr Anwendungsnachrichten, jedoch sollte die maximale Zeit in der eine Nachricht blockiert wird verringert werden. Zeitgleich ist auch die Gefahr der volllaufenden Puffer und der damit verbundene Rückstau geringer, als es bei der XY-Strategie der Fall ist. Die nachfolgende Untersuchung soll den Vorteil von Staircase-Routing gegenüber XY-Routing quantitativ bewerten.

3.3. Einfluss der Heartbeat-Nachrichten auf Anwendungsnachrichten

Die Last, die ein Netz bewältigen kann, hat einen starken Einfluss auf die gesamte Leistungsfähigkeit eines Prozessors. Ist das Netz unterdimensioniert, kann dies schnell zu langen Zeiträumen führen, in denen ein Prozessor unproduktiv auf eine Antwort wartet. Verstärkt werden solche Wartezeiten für Anwendungsnachrichten, durch die zuvor beschriebene Priorisierung der Heartbeat-Nachrichten. Da die Heartbeat-Nachrichten die höchste Priorität im Netz haben, wird erwartet, dass Anwendungsnachrichten länger für den Weg durch das Netz benötigen. Diese Wartezeiten sind - wenn man die Antwortzeit des Kommunikationspartners nicht beachtet - im Allgemeinen von den Kennzahlen *Durchsatz*, *Latenz* und *Jitter* abhängig. Die Definitionen und Erklärungen der Evaluierungsmetriken basieren, wenn nicht anders angegeben, auf der Fachliteratur von [Duato 2002; Dally und Towles 2003].

3.3.1. Metriken der Evaluierung

Der Durchsatz beschreibt die maximale Datenmenge, die das Netz aufnehmen und verarbeiten kann[6]. Werden über den Durchsatz hinaus weitere Daten in das Netz emittiert, *betreten* mehr Nachrichten das Netz, als aus dem Netz *abfließen* können und es kommt zur sogenannten Sättigung des Netzes. Sobald die maximale Bandbreite einer Verbindungsleitung überstiegen wird, beginnen sich die Puffer des Routers an dieser Leitung stetig zu füllen. Sind die Puffer voll, kommt es nicht nur zu Staus am betroffenen Router. Stattdessen führt dies auch zu einem Rückstau bis hin zu den emittierenden Prozessorkernen. Dabei ist es nicht zwingend notwendig, dass alle Prozessorkerne übermäßig viele Daten in das Netz entsenden müssen, um die Sättigung des Netzes zu erreichen. Der Durchsatz eines Netzes hängt stattdessen sehr von dem zugrundeliegenden Sendemuster und dessen *Injektionsrate* ab. Ein ähnliches Phänomen sind die durch Heartbeat-Nachrichten besonders stark belasteten Verbindungsleitungen in der Nähe der FDU. Hier sorgt die hohe Zahl der Heartbeat-Nachrichten wegen ihrer hohen Priorität im Netz für eine Verringerung der effektiven Bandbreite. Für Anwendungsnachrichten bedeutet dies potentiell längere Wartezeiten in diesem Bereich des Netzes und damit auch einen geringeren Durchsatz der Nutzdaten.

Die Latenz misst die Zeit von der Initiierung der Datenübertragung bis zum vollständigen Empfang der Daten. Die Latenz ist, neben der Distanz zwischen Sender und Empfänger, maßgeblich von der vorherrschenden Last im Netz abhängig. Auch hier gilt, dass vor allem das Sendemuster in Verbindung mit der Injektionsrate die Last des Netzes definiert. Bei wenig Last können die Daten nahezu ohne Verzögerung übermittelt werden

[6] In der Literatur auch als akzeptierte Last (*accepted traffic*) bezeichnet

und die Latenz wird damit nur von der Router-Pipeline und den Verzögerungen der Verbindungsleitungen dominiert. Strebt die Last des Netzes hingegen gegen den maximalen Durchsatz, besteht die Möglichkeit, dass durch Verzögerungen[7] im Netz die Latenz stark ansteigt. Wie erwähnt, füllen sich die Puffer der Router beim maximalen Durchsatz zum Teil bis zum Sender, sodass in diesen Fällen die Latenz theoretisch unendlich wird. Daher ist eine Untersuchung der Latenz unter Berücksichtigung des Sendemusters und der Injektionsrate jenseits des maximalen Durchsatzes unnötig. Zusätzlich muss sichergestellt werden, dass die Puffer des Senders (nicht aber im Netz selbst) groß genug sind, um blockierte Nachrichten zwischenspeichern zu können. Andernfalls sorgt ein gefüllter Puffer dafür, dass der Sender keine neuen Nachrichten erzeugen kann, oder diese verwirft, weil kein Platz mehr im Puffer ist. Dies würde jedoch das eigentliche Verhalten des Sendemuster verfälschen (siehe weiter unten). Daher wird die Kapazität des Ausgangspuffers jedes Senders in der Simulation unendlich groß dimensioniert [Duato 2002; Dally und Towles 2003].

Unglücklicherweise gibt es bisher keinen einheitlichen Standard in Bezug auf Sendemuster zur Evaluierung eines Kommunikationsnetzes innerhalb eines Prozessors. Zwar wird bereits an einer Standardisierung gearbeitet [Salminen 2010; Pekkarinen 2011], dennoch existieren die Benchmarks nur für spezielle Probleme oder setzen eine Full-System-Simulation voraus. Die hier angestrebte Untersuchung hingegen befasst sich ausschließlich mit den Kennzahlen Durchsatz, Latenz und Jitter und kann daher mit einer Reihe synthetischer Sendemuster durchgeführt werden. Diese Sendemuster werden durch die folgenden drei Parameter definiert und auf die sendenden Prozessorkerne angewendet: *Verteilung der Empfänger*, *Injektionsrate* und *Länge der Nachrichten*.

Der Jitter ist eine wichtige Kennziffer um die gleichmäßige Geschwindigkeit der Nachrichtenverarbeitung eines Netzes zu bestimmen. Ist der Jitter in einem Netz hoch, kann es dazu kommen, dass Anwendungen, welche einen konstanten und gleichmäßigen Datenstrom erwarten, durch hohe Jitter-Werte ins "Stocken" geraten. Gerade bei Ausführungsmodellen bei denen Programmteile auf verschiedenen Prozessorkernen in einer Art Pipeline ausgeführt werden, wirken Jitter-Effekte merklich, da in so einem Fall der gesamte Programmfortschritt an einer Pipeline-Stufe verzögert wird. Ein Beispiel, bei dem der Benutzer hohe Jitter schnell bemerkt, ist das Video-Dekodieren. Tritt hier ein hoher Jitter auf, kann es leicht zu dem bekannten "Bild-Ruckeln" kommen, welcher der das Bild kurzzeitig zum Stillstand bringt.

Um den Jitter des Netzes hinsichtlich des verwendeten Sendemusters innerhalb der Simulationsumgebung zu ermitteln, können zwei Hilfsmittel herangezogen werden. Zum Einen kann mit der Berechnung der Standardabweichung der durchschnittlichen Latenzzeiten ermittelt werden, wie stark die Latenzzeiten fluktuieren. Eine hohe Fluktuation ist dabei ein Indikator für einen hohen Jitter. Außerdem können die maximalen Latenzen

[7] Ausgelöst durch konkurrierenden Zugriff auf die Netzressourcen.

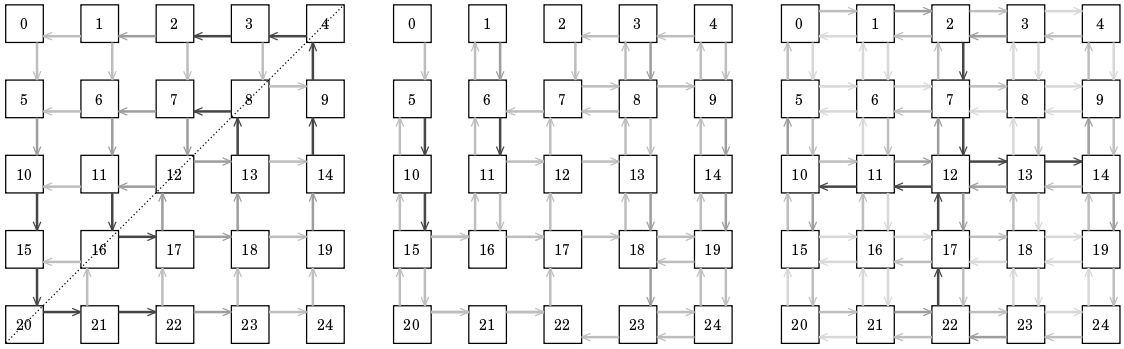
der Anwendungsnachrichten gemessen werden. Sie geben eine gemessene obere Schranke des Jitters an und stellen den "worst case" dar.

Die Verteilung der Empfänger definiert für einen gegebenen Sender den nächsten Empfänger seiner Nachricht. Sie stellt gewissermaßen die räumliche Verteilung der Kommunikationspartner dar und wird als *Sendemuster für Anwendungsnachrichten* bezeichnet. Die am häufigsten gewählte Verteilung ist die Gleichverteilung. Jeder Prozessorkern besitzt also die gleiche Wahrscheinlichkeit als Empfänger einer Nachricht ausgewählt zu werden. Daher eignet sich dieses Muster gut, um eine nahezu gleichmäßige Lastverteilung im Netz zu erzeugen. Das gilt selbst für Topologien und Routing-Strategien, die an sich keine adaptive Lastverteilung unterstützen. Dies ist insbesondere dann von Interesse, wenn der Evaluierung mit einem Simulator eine analytische Berechnung des Durchsatzes und der Latenz voran gegangen ist. Die Berechnungen werden aus Vereinfachungen heraus mit gleichverteilten Werten durchgeführt und gehen noch nicht auf spezielle Parameter des Netzes wie der Puffergröße ein. Damit erreicht man schnell eine erste grundsätzliche Evaluierung. Beachtet man zusätzlich, dass der maximale Durchsatz und die Latenz stärker durch die Last im Netz, als durch Implementierungsparameter beeinflusst werden, eigenen sich die errechneten Ergebnisse auch zur Kontrolle, ob die Simulation das Netz akkurat modelliert.

Die weiteren Sendemuster der Anwendungsnachrichten basieren auf statischen Zuordnungen, bei denen im Vorfeld der Simulation zu jedem Sender genau ein Empfänger zugeordnet wird. Die Kommunikationspaare werden anhand ihrer Adress-Bits durch Bit-Permutationen ermittelt und festgelegt. Durch die Permutationsverfahren, wie dem *Transpose*, *Shuffle* oder *Bit-Reversal*, gelingt es charakteristische Sendemuster für Anwendungsnachrichten zu erzeugen, welche häufig bei numerischen Algorithmen auftreten [Kim und Chien 1992; Duato 2002]. Für die folgende Evaluierung wurden die Sendemuster Transpose und Bit-Reversal ausgewählt. Zusätzlich wurde ein Hot-Spot Muster hinzugefügt, das, neben dem der FDU, ein zusätzliches Narrow-Cast-Muster für Anwendungsnachrichten implementiert. Hier werden ein Teil aller Anwendungsnachrichten an verschiedene am Rand des Prozessors liegende Controller[8] gesendet. Der verbleibende Anteil der Kommunikation findet unterdessen mit dem oben angeführten gleich verteilten Muster statt. Die drei Sendemuster wurden in der Abbildung 3.7 zusammengefasst. Die Pfeile in den einzelnen Abbildungen zeigen die verwendeten Verbindungsleitungen und die Senderichtung an. Der Grad der Schattierung der Pfeile gibt an, wie stark die erwartete durchschnittliche Belastung einer bestimmten Verbindungsleitung allein durch Anwendungsnachrichten ist.

Die Bezeichnung "Transpose" leitet sich vom Transponieren einer Matrix ab. In einem Netz N wird ein Sender $N_{x,y}$ mit dem Empfänger $N_{x,y}^T$ fest verbunden. Die Simulationsumgebung des Noxim [Palesi 2007] transponierte die Netzkoordinaten über

[8] Klassischerweise sind dies Controller für entfernte Speicher oder Ein-/Ausgabe-Geräte.



(a) Transpose (b) Bit-Reversal (c) Hot-Spot

Abb. 3.7.: Schematische Darstellung der verwendeten Sendemuster für Anwendungsnachrichten.

die in Abbildung 3.7(a) eingezeichnete gestrichelte Linie. Interessanterweise tritt diese Sender-Empfänger-Zuordnung tatsächlich bei Operationen der Matrix Transponierung oder beim Corner-Turn auf. Mit der Corner-Turn Operation werden Daten so umorganisiert, sodass beispielsweise Bildverarbeitungsalgorithmen effizienter durch Datenparallelität auf Mehrkern-Systemen ausgeführt werden können [Wentzlaff 2007]. Die dunklen Pfeile in der Abbildung zeigen, dass ein großer Anteil der Awendungsnachrichten an den Randbereichen, sowie in mittlerer Distanz zur FDU, durch das Netz geleitet wird. Das Bit-Reversal Muster tritt meist bei Fast Fourier Transformationen auf, um ebenfalls die Daten effizient zu organisieren. In Abbildung 3.7(b) ist die Lastverteilung für Bit-Reversal schematisch dargestellt. Hier ist zu erkennen, dass das Sendemuster überwiegend Kommunikation in vertikaler Richtung erzeugt. Zudem konzentriert sich ein Teil der Belastung durch Anwendungsnachrichten an den Routern mit den IDs 5,6 und 10.

Abschließend ist in Abbildung 3.7(c) das Hot-Spot Sendemuster für Anwendungsnachrichten dargestellt. Die Elemente mit den IDs 2,5,10 und 22 nehmen in diesem Sendemuster die Rolle von Speichercontrollern ein. Die Controller werden von den übrigen Prozessorkernen mit einer Wahrscheinlichkeit von 60% als Ziel ausgewählt. Erhält ein Controller eine Nachricht, wird auf diese im nächsten Takt direkt geantwortet. Da innerhalb dieser Evaluierung nicht die Leistungsfähigkeit des Prozessors in Bezug auf die Anwendungsausführung getestet wird, ist die Antwortzeit des Controllers nicht kritisch und kann mit der Dauer von einem Takt angenommen werden. Der Zweck dieses Hot-Spot Musters liegt darin, einen Großteil der Kommunikation durch das Zentrum des Netzes zu leiten. Das Hot-Spot Muster bietet sich daher für diesen Zweck an.

Die Injektionsrate ist die zeitliche Verteilung der Nachrichtenübermittlung. Auch hier existieren verschiedene Muster, bei denen der Zeitpunkt für die Initiierung einer Kommunikation bestimmt wird. Ein gängiges Muster ist, ähnlich wie bei der räumlichen

Verteilung, die Normalverteilung mit der Erweiterung einer Bernoulli-Verteilung um Kommunikations-Bursts zu simulieren [Duato 2002].

3.3.2. Aufbau der Simulationsumgebung

Die Simulationen decken unterschiedliche Versuchsaufbauten hinsichtlich des Einsatzes von Heartbeat-Nachrichten unter Verwendung der verschiedenen Routing-Strategien ab:

- Grundlagenmessung ohne Heartbeat-Nachrichten.
- Mit Heartbeat-Nachrichten:
  - XY-Strategie (R_{XY})
  - Staircase-Strategie (R_{ST})

Die Anwendungsnachrichten werden unterdessen stets mit der XY-Strategie durch das Netz geleitet.

Simuliert wurde die Kommunikationsstruktur eines Prozessors mit 64 Kernen. Die Kerne selbst bestehen hierbei nur aus einem Empfangs- und einem Sende-Modul, mit denen Nachrichtenpakete empfangen, beziehungsweise gesendet werden können. Welcher Kern mit wem kommuniziert und zu welchem Zeitpunkt, wird für Anwendungsnachrichten durch das in der Simulation verwendete Sendemuster und die Injektionsrate entschieden. Hierfür wurden Transpose, Bit-Reversal, das Hot-Spot-Muster und das normalverteilte Zufallsmuster eingesetzt. Die tatsächliche Ausführung eines Programms wird nicht unterstützt und ist nicht Teil der Evaluierung. Jedoch nehmen die Kerne im Zuge der Evaluierung verschiedene Rollen an, die durch das verwendete Sendemuster festgelegt werden:

- FDU: Ein spezieller Kern, der in der Lage ist Heartbeat-Nachrichten an andere Kerne zu übermitteln um das Polling zu implementieren, bzw. die Kerne mit dem TDMA-Schema zu konfigurieren. Die FDU versendet darüber hinaus keine Anwendungsnachrichten.
- Asynchron: Die Prozessorkerne konsumieren Anwendungsnachrichten, jedoch beantworten sie diese nicht direkt. Dies kann zur Simulation von Lasten verschiedener Ausführungsmodelle, wie dem Pipelining und Fork-Join, eingesetzt werden.
- Synchron: Die Prozessorkerne konsumieren Anwendungsnachrichten und beantworten diese. Dies simuliert das Verhalten eines Client-Server-Ausführungsmodels oder den Zugriff auf ein anderes Gerät des Chips (Speicher-Controller oder Ein-/Ausgabe-Controller).

Zusätzlich erzeugen und versenden die Prozessoren, gemäß dem TDMA-Schema, die Heartbeat-Nachrichten für die FDU.

Jeder Prozessorkern ist mit genau einem Router verbunden und nutzt diesen zur Kommunikation mit anderen Komponenten im Netz. Die Kerne sind dazu mit einer bidirektionalen Verbindungsleitung an den Router gekoppelt, was eine Full-Duplex Kommunikation zwischen Kern und Router ermöglicht. Die gleiche Kopplung wurde ebenfalls für die Verbindung zwischen den Routern eingerichtet, wobei die Topologie der verbundenen Router einem zweidimensionalen, homogenen Gitter entspricht.

Router-intern endet jede eingehende Verbindungsleitung an einer Demux-Einheit, die anhand eines Paket-Flags die eingehenden Nachrichten auf zwei Eingangspuffer aufteilt. Grundlage der Aufteilung ist die räumliche Isolation von Anwendungsnachrichten und Heartbeat-Nachrichten. Zusätzlich wird, unabhängig vom Paket-Flag, die Routing-Logik direkt bei Ankunft einer Nachricht angewendet. Da die Routing-Logik der XY-Strategie und der Staircase-Strategie vergleichsweise leichtgewichtig ist, kann angenommen werden, dass diese noch im selben Netztakt das Ergebnis für die Paketweiterleitung liefert. Die Flußkontrolle basiert auf einem zweistufigem Wormhole-Switching, welches für das QoS erweitert wurde. Die Arbiter-Einheit prüft dazu zunächst den Inhalt der Puffer für Pakete hoher Priorität (Heartbeat-Nachrichten) und reserviert gleichzeitig für eine gegebene Nachricht die Ein-/Ausgabe-Verbindung der Router-internen Kreuzschiene. In der zweiten Stufe wird dieser Vorgang für die Nachrichten geringer Prioritäten wiederholt. Auch hier kann angenommen werden, dass die Kanal-Arbitrierung und Weiterleitung der Pakete in einem Takt vollzogen werden. Somit ergibt sich für die Simulation eine Router-Pipeline, die in zwei Takten von einer Heartbeat-Nachricht durchlaufen wird. Anwendungsnachrichten benötigen aufgrund ihrer variablen Größe von 2 bis 4 Flits zwischen 4 und 8 Takte. Dies setzt jedoch voraus, dass die Nachricht nicht von einer anderen Nachricht, durch konkurrierenden Zugriff auf die gleiche Ausgangsleitung, blockiert wird.

Die Sendemuster für Anwendungsnachrichten in den Simulationen wurden mit einer Reihe verschiedener Paket-Injektionsraten als Simulationsparameter ausgeführt. Diese Injektionsraten kontrollieren die Netzlast, die von Anwendungsnachrichten erzeugt werden und variieren von $1e^{-5}$ bis zur theoretisch maximalen Netzkapazität eines gegebenen Sendemusters. Das Ziel der verschiedenen Injektionsraten war, den Sättigungspunkt des Netzes zu ermitteln. Als Indikator für eine Sättigung wird der durchschnittliche Durchsatz über alle Flits ermittelt. Steigt dieser ab einer bestimmten Injektionsrate nicht weiter an, signalisiert dies, dass mehr Flits erzeugt werden, als das Netz in der Lage ist auszuliefern. Die Folge ist ein stagnierender Durchsatz auch bei höheren Injektionsraten. Jenseits der Sättigungsgrenze erzeugen steigende Injektionsraten vor allem aber stark steigende Latenzen bei den Flits. Diese werden ab diesem Zeitpunkt bereits direkt nach der Erzeugung in den Upload-Puffer des Prozessorkerns geladen und verweilen dort so lange, bis der angeschlossene Router diese von dort verarbeiten kann. Da ein voller Puffer dafür sorgen würde, dass der Sender keine neuen Nachrichten erzeugen kann und damit das Sendemuster verändert, wird zur Vereinfachung ein unendlich großer Puffer für die Sender implementiert [Duato 2002; Dally und Towles 2003]. Neben den Verzögerungen innerhalb des Netzes kommt es also bereits vor dem Eintritt in das Netz zu Verzögerun-

gen. Eine weitere Erhöhung der Injektionsrate über den Sättigungspunkt hinaus macht also bei der Untersuchung hinsichtlich der Latenz keinen Sinn.

Zusätzlich zu den Sendemustern kamen zur Lasterzeugung drei verschieden enggestaffelte TDMA-Schemata der Heartbeat-Nachrichten zum Einsatz. Das Schema $x1$ erzeugt das engste Heartbeat-Muster und sorgt somit für die stärkste Belastung auf den Verbindungsleitungen in FDU-Nähe. Zudem wurden mit den Schemata $x2$ (Belastung halbiert) und $x4$ (Belastung geviertelt) weniger enggestaffelte Muster erzeugt.

3.3.3. Durchsatz

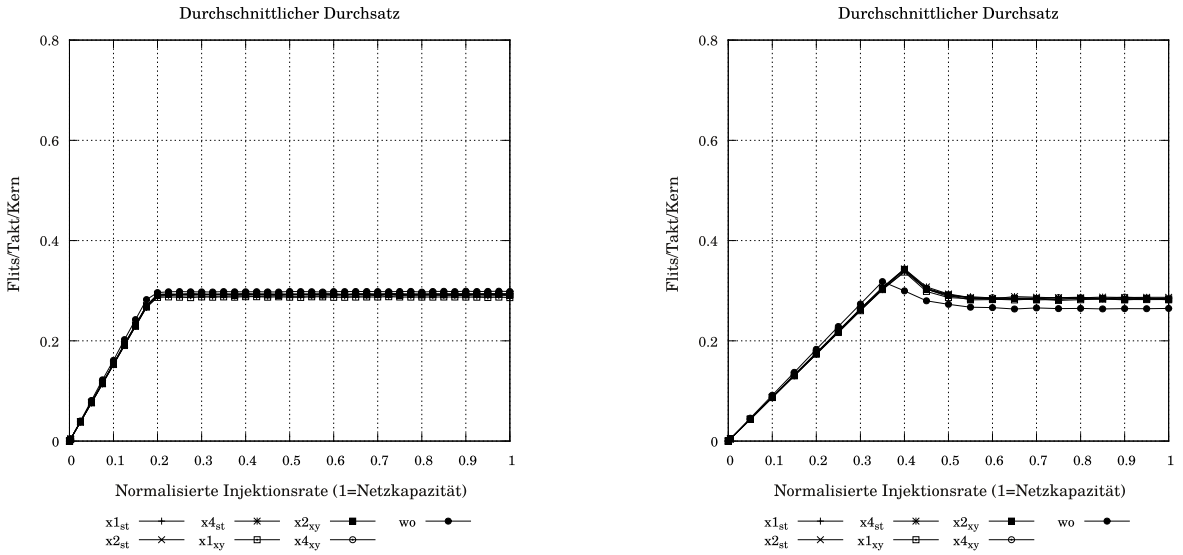
Wie bereits beschrieben, ist der Durchsatz unter Verwendung des jeweiligen Sendemusters ein wichtiger Indikator für die Leistungsfähigkeit eines Kommunikationsnetzes. So zeigt dieser bei stagnierenden Werten den Sättigungspunkt des Netzes an und somit die obere Schranke für die spätere Untersuchung hinsichtlich der Latenzen und des Jitters. Für die Untersuchung wurde der Simulator zu einem 5×5 großen Netz konfiguriert und für Anwendungsnachrichten mit den Sendemustern Random, Transpose, Bit-Reversal und Hot-Spot ausgeführt. Insgesamt wurden zu jedem Muster drei Versuchsreihen durchgeführt, bei denen zweimal jeweils Heartbeat-Nachrichten mit der XY- (R_{XY}) oder der Staircase-Strategie (R_{ST}) übertragen wurden. Die dritte Versuchsreihe wurde ohne Heartbeat-Nachrichten durchgeführt und dient als Vergleichsgrundlage der Ergebnisauswertung.

Um die verschiedenen Sendemuster auch untereinander in Relation zu setzen, wurden die Injektionsraten ins Verhältnis zur jeweiligen Netzkapazität[9] normalisiert. Die Abbildung 3.8(a) fasst die Simulationsergebnisse hinsichtlich des Durchsatzes für die Sendemuster Random und Hot-Spot zusammen. Zur besseren Darstellung und Vergleichbarkeit wurden die Injektionsraten (X-Achse) auf die jeweiligen Verteilungsmuster normalisiert. Hierzu diente die obere Grenze des theoretisch erreichbaren Durchsatzes für dieses Netz unter dem gegebenen Verteilungsmuster. Auf der Y-Achse ist die Menge der ablaufenden Flits pro Netztakt und Prozessorkern aus dem Netz aufgetragen.

Beim Sendemuster Random zeigt das Netz nach dem Erreichen des Sättigungspunktes ein konstant stagnierendes Durchsatzvermögen. Man spricht hierbei auch von einem stabilen Netz nach Erreichen des Sättigungspunkts. Beim Sendemuster Hot-Spot hingegen zeigt sich ein deutlicher Abfall des Durchsatzes von ≈ 0.08 auf ≈ 0.07, der sich jedoch mit steigendender Injektionsrate wieder stabilisiert. Der Abfall des Durchsatzes begründet sich darin, dass etwa 60% der gesamten Kommunikation[10] direkt auf den Verbindungsleitungen verarbeitet werden müssen, welche bereits stark durch die Heartbeat-Nachrichten

[9] Die Netzkapazität ist der theoretisch maximal erreichbare Durchsatz eines Netzes unter Berücksichtigung des verwendeten Sendemusters.

[10] Die vier Hot-Spot Punkte werden jeweils mit einer Wahrscheinlichkeit von 15% als mögliches Kommunikationsziel ausgewählt.



(a) Lastverteilung: Random (b) Lastverteilung: Hot-Spot

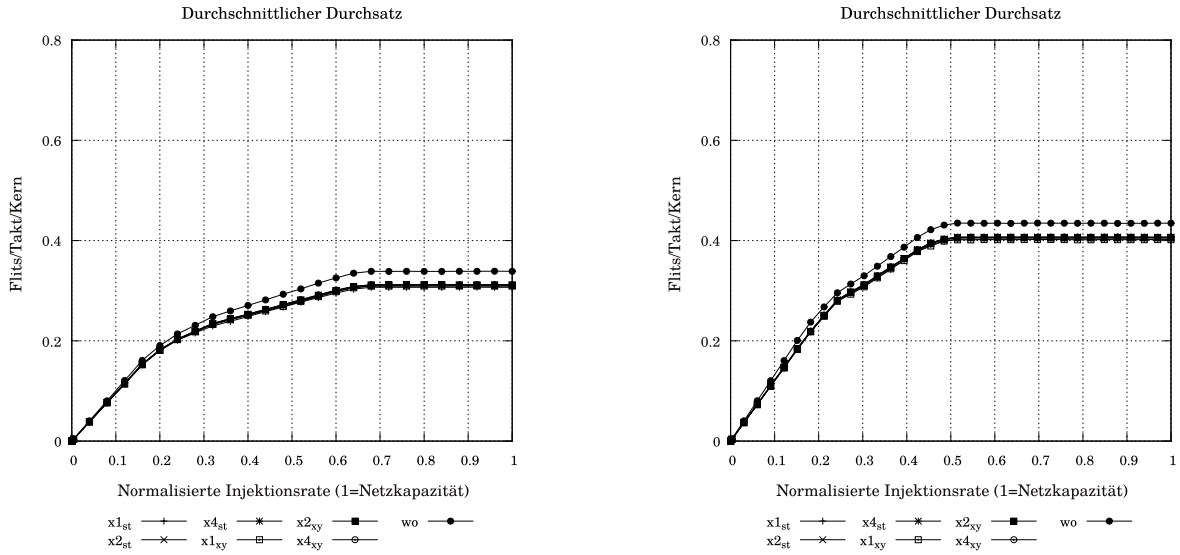
Abb. 3.8.: Durchschnittlicher Durchsatz gemessen für die Lastverteilungen Random und Hot-Spot.

belastet sind. Zusätzlich kommt hinzu, dass durch das QoS keine Fairness bei der Behandlung zwischen den Heartbeats und den Anwendungsnachrichten besteht. Die beiden Punkte zusammengenommen ergeben für dieses Sendemustermuster den Verlust des Durchsatzvermögens nach dem Sättigungspunkt.

Da beim Sendemuster Random alle Kommunikationspaare durch eine gleichverteilte Wahrscheinlichkeit ermittelt werden, ist auch die Belastung des Netzes gleichverteilt. Dies sorgt dafür, dass das Netz auch nach dem Sättigungspunkt stabil bleibt.

In Abbildung 3.9 sind die Simulationsergebnisse der Sendemuster Transpose und Bit-Reversal aufgetragen. In beiden Sendemustern zeigt das Netz einen stabilen Durchsatz, auch nach dem Erreichen des Sättigungspunktes. Deutlich zu erkennen ist, dass das Netz für das Kommunikationsmuster Bit-Reversal den höchsten Durchsatz aller vier getesteten Muster erzielt, was überwiegend an der Lage der Kommunikationspaare liegt. Sender und Empfänger liegen bei diesem Muster jeweils diagonal auf der anderen Seite des Netzes. Daraus resultiert ein kreisförmiger Weg, den die Nachrichten eines Kommunikationspaares beschreiben. Zusätzlich bewegen sich die Nachrichten immer in gleicher Richtung, was die Wahrscheinlichkeit eines Blockierens verringert.

Jedoch “umkreisen” die Nachrichten bei beiden Routing-Strategien das Zentrum des Netzes. Bei den Ergebnissen aus den beiden Simulationsläufen mit Heartbeat-Nachrichten, resultiert eine Verschlechterung des Durchsatzes um etwa 0.03 Flits pro Netztakt und Prozessorkern. Obwohl der Einfluss der Heartbeat-Nachrichten marginal auf die durch-



(a) Lastverteilung: Transpose

(b) Lastverteilung: Bit-Reversal

Abb. 3.9.: Durchschnittlicher Durchsatz gemessen für die Lastverteilungen Transpose und Bit-Reversal.

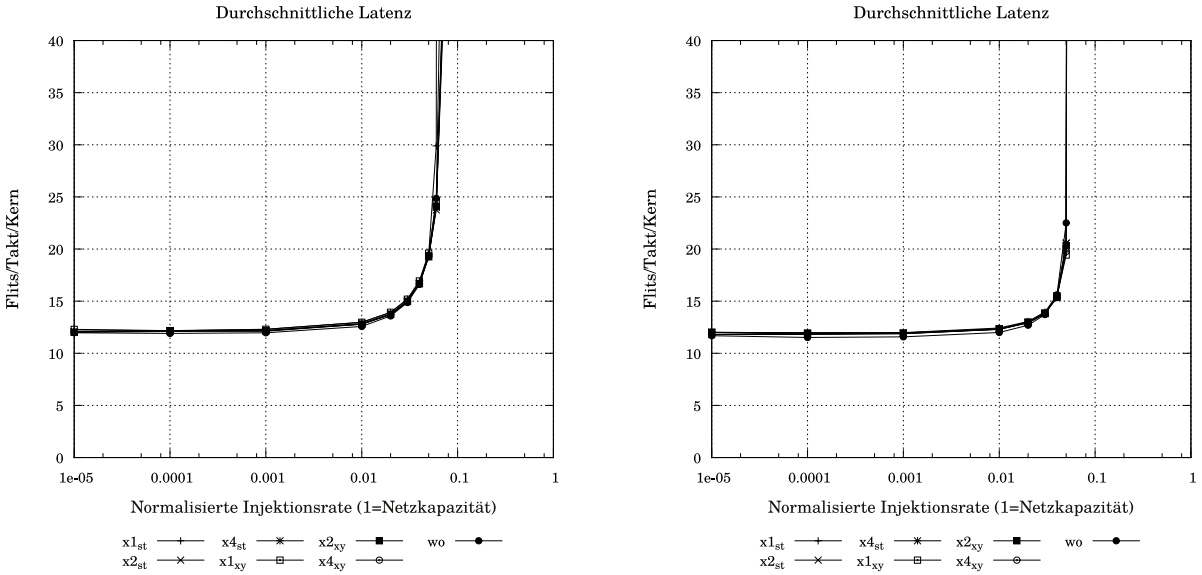
schnittlichen Durchsatz ist, zeigt sich dennoch das erste Mal eine messbare Änderung der Leistungsfähigkeit des Netzes.

3.3.4. Latenzzeiten

Neben der Fähigkeit möglichst viele Pakete parallel übertragen zu können, ist auch die Geschwindigkeit, in der ein einzelnes Flit durch das Netz verarbeitet wird, entscheidend für die Leistungsfähigkeit des Netzes. Ohne den Einsatz der Heartbeat-Nachrichten liegt die erwartete durchschnittliche Latenz Λ_{ideal} in dieser Netzkonfiguration bei 12 Takten[11] [Dally und Towles 2003]. Die Abbildungen 3.10(a und b) sowie 3.11(c und d) zeigen die Resultate der Simulation bezüglich der Latenz von Anwendungsnachrichten. Zusätzlich wurde aus Gründen der besseren Darstellung die x-Achse logarithmisch skaliert. Damit wird ein etwas detaillierterer Ausschnitt der Latenzwerte ermöglicht.

Leicht erkennbar ist, dass alle Graphen eng zusammenliegen und man somit, ähnlich wie beim Durchsatz, auf einen durchschnittlich geringen Einfluss der Heartbeat-Nachrichten auf Anwendungsnachrichten schließen kann. Außerdem liegen die Werte der Latenz bei allen angewendeten Kommunikationsmustern nahe dem theoretischen Minimum Λ_{ideal}, was ebenfalls positiv zu werten ist. Selbst unter der Verwendung von enggestaffelten Heartbeat-Nachrichten ($x1_{xy}$ und $x1_{st}$) sind keine signifikanten Unterschiede feststellbar. Während diese Beobachtung überwiegend bei geringeren Injektionsraten zu erwar-

[11] Λ = Pipeline-Länge $\times$ $(H_d + Drain) + 1$, mit H_d als durchschnittliche Pfadlänge in Hops



(a) Lastverteilung: Random

(b) Lastverteilung: Bit-Reversal

Abb. 3.10.: Durchschnittliche Latenz gemessen für Random und Bit-Reversal.

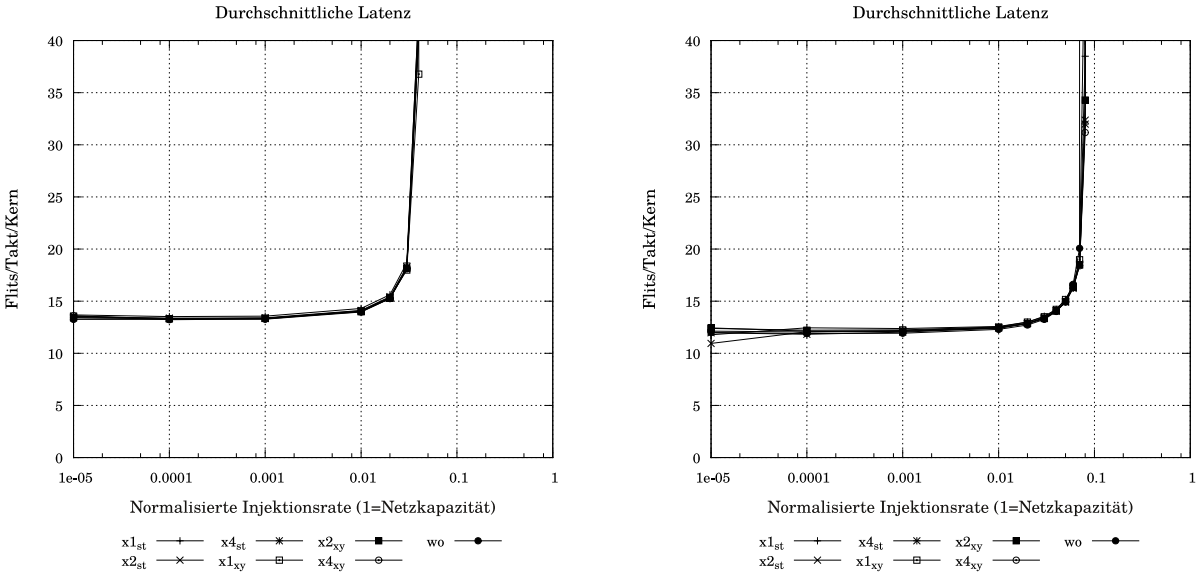
ten war, ist die gleichmäßige Entwicklung der Latenzzeiten nahe dem Sättigungspunkt des Netzes ungewöhnlich, da bei diesen Injektionsraten ein deutlicher Rückstau der Anwendungsnachrichten zu erwarten ist und auch auf andere Anwendungsnachrichten einen Einfluss ausüben sollte.

Der Grund für diese dennoch gleichmäßige Latenzentwicklung kann durch die Routerinternen Eingangspuffer beantwortet werden. Die Dimensionierung der Eingangspuffer entspricht der maximalen Länge der Anwendungsnachrichten. Das bedeutet also, dass eine blockierte Nachricht vollständig vom Eingangspuffer eines Routers aufgenommen werden kann und somit die Nachricht nur die Ressourcen eines einzelnen Routers belegt. Das Resultat ist, auch bei Injektionsraten nahe dem Sättigungspunkt, insgesamt eine Verringerung der Wahrscheinlichkeit für Rückstaus. Der Vollständigkeit halber sei erwähnt, dass weitere (gleichberechtigte[12]) virtuelle Kanäle für Anwendungsnachrichten auch bei größeren Nachrichtlängen den Effekt des Rückstaus verringern können.

3.3.5. Jitter

Die Untersuchung des Jitters wird differenzierter hinsichtlich des Sendemusters, der Injektionsrate und der Pfadlänge durchgeführt. Die Ergebnisse der Untersuchung basieren dabei auf der Messung der maximalen Verzögerung, welche die Nachrichten während der Simulation ausgesetzt waren. Dazu wurden die gemessene Laufzeit und die Pfadlänge durch das Netz einer Nachricht bestimmt. Mit Hilfe der Pfadlänge $|p|$ kann die gemessene

[12]In Bezug auf eine faire Arbitrierung innerhalb des Routers.



(a) Lastverteilung: Transpose (b) Lastverteilung: Hot-Spot

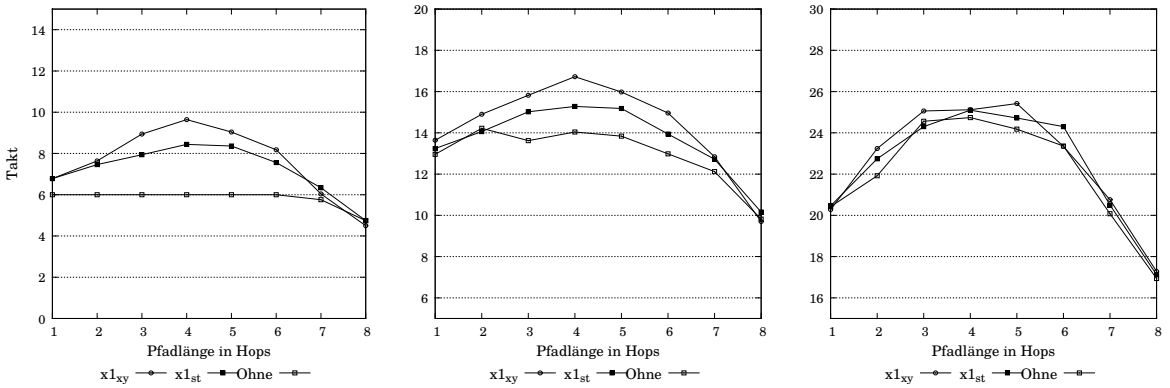
Abb. 3.11.: Durchschnittliche Latenz gemessen für die vier Lastverteilungen Transpose und Hot-Spot.

Latenz von Λ_{ideal} bereinigt werden und ergibt somit die Zeit der Verzögerung $\phi_{|p|}$ in der die Nachricht keinen Fortschritt im Netz machen konnte. Abschließend wurde für jede Pfadlänge $|p|$ die maximale Verzögerung $\phi_{(max,|p|)}$ bestimmt.

Den Ergebnissen vorweg genommen sei erwähnt, dass bei allen Sendemustern, Nachrichten mit einer mittleren Pfadlänge von 3−6 Hops stärker von den Verzögerungen betroffen sind als bei den Längen $1-2$ und $7-8$. Dieser Effekt hat zwei Gründe. Zum einen bewegen sich Nachrichten mit kurzen Pfaden ($1-2$ Hops) nur kurz im Netz. Damit ist im Durchschnitt die Wahrscheinlichkeit der mehrfachen Verzögerung der Anwendungsnachrichten durch Heartbeat-Nachrichten geringer als bei längeren Pfaden. Bei den Pfaden der Länge $7-8$ sind Anwendungsnachrichten zwar länger im Netz, jedoch erzeugt die XY-Strategie Pfade, die am Rand des Netzes verlaufen. Aus Abbildung 3.5(a) und 3.6(a) ist ersichtlich, dass die Belastung durch Heartbeat-Nachrichten gerade in diesen Randbereichen des Netzes gering sind und somit auch die Wahrscheinlichkeit einer mehrfachen Verzögerung der Nachrichten.

Die Abbildungen 3.12(a)-(c) stellen den jeweiligen Verlauf von $\phi_{(max,|p|)}$ für das Sendemuster Random mit verschiedenen Injektionsraten und aufsteigend nach $|p|$ sortiert dar. Die x-Achse unterteilt die Ergebnisse in die jeweiligen Pfadlängen. Auf der y-Achse ist die maximale Verzögerung $\phi_{(max,|p|)}$ aufgetragen, wobei eine Verzögerung mit einem Wert = 0 dem Λ_{ideal} entspricht.

Bei geringen Injektionsraten (siehe Abbildung 3.12(a)) ist ein konstanter Wert der maximalen Verzögerungen bei der Simulation ohne Heartbeat-Nachrichten zu erkennen. Es



(a) Injektionsrate: 0.00001 (b) Injektionsrate: 0.001 (c) Injektionsrate: 0.01

Abb. 3.12.: Maximale Latenz gemessen für die Lastverteilung Random.

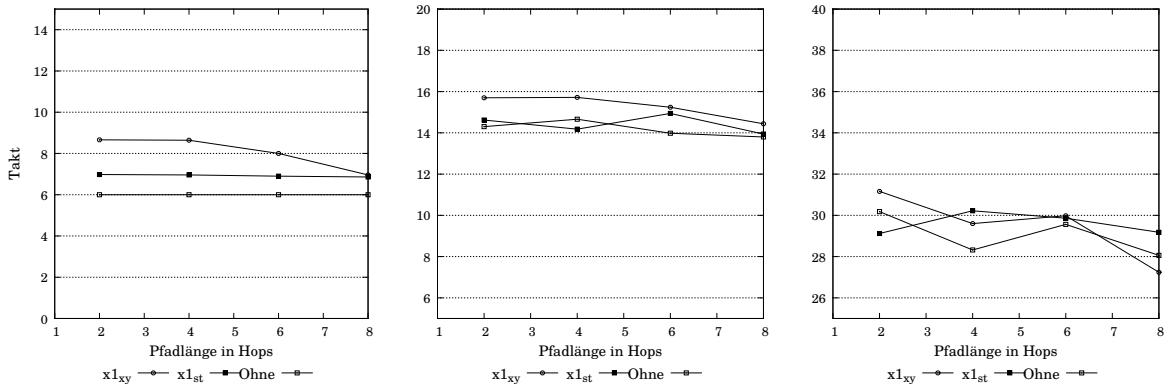
zeigt sich also, dass das Netz gleichmäßig mit Anwendungsnachrichten belastet wird. Die leichte Verringerung ab einer Pfadlänge von 7 Hops kann durch das oben erwähnte Verhalten des Routings von Nachrichten mit langen Pfadlängen erklärt werden.

Deutlich längere Verzögerungen sind dagegen bei den Ergebnissen der Simulationen mit Heartbeat-Nachrichten ($x1_{xy}$ und $x1_{st}$) zu erkennen. Wird die XY-Strategie für Heartbeat-Nachrichten verwendet, liegt die maximale Verzögerung im Netz um etwa 60% höher, als beim Referenz-Versuch ohne Heartbeat-Nachrichten. Bei der Staircase-Strategie steigt die Wartezeit im maximalen Fall um etwa 40%.

Mit steigender Injektionsrate, steigt auch die Menge der Anwendungsnachrichten im Netz. Die Folge sind häufigere Kollisionen der Anwendungsnachrichten untereinander, womit auch die maximalen Verzögerungen in den Referenz-Simulationen steigen. Dies lässt sich anhand der Abbildung 3.12(b) gut beobachten. Die maximale Verzögerung verzeichnet einen Zuwachs von vormals 6 Takten auf nun 14 Takte. Des Weiteren ist zu erkennen, dass die Werte der maximalen Verzögerung mit Heartbeat-Nachrichten nicht in dem Maße wachsen, wie es bei der Referenz-Simulation der Fall ist. Die maximale Verzögerung liegt jetzt bei $\approx 20\%$ (XY-Strategie) und $\approx 13\%$ (Staircase-Strategie) über der Referenz-Simulation. Das bedeutet, die Werte der maximalen Verzögerung werden zunehmend von den Kollisionen der Anwendungsnachrichten untereinander dominiert.

Kurz vor dem Sättigungspunkt des Netzes sind keine signifikanten Unterschiede zwischen den Routing-Strategien erkennbar (Abbildung 3.12(c)). Auch das Ergebnis der Simulation ohne Heartbeat-Nachrichten weicht nicht mehr signifikant von den anderen ab. Das bedeutet, der Anteil an Kollisionen zwischen Anwendungsnachrichten und Heartbeat-Nachrichten ist auf ein Minimum gesunken und trägt somit nicht erkennbar zu den maximalen Verzögerungen bei.

Wie bereits weiter oben erwähnt, ist das Sendemuster Random zwar ein gutes Maß um Netze mit gleichmäßiger Lastverteilung im gesamten Netz zu testen, doch entspricht eine



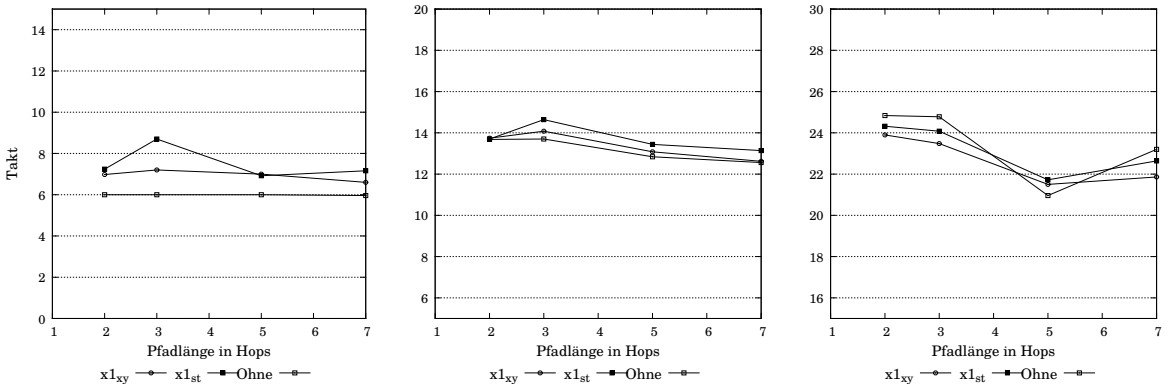
(a) Injektionsrate: 0.00001 (b) Injektionsrate: 0.001 (c) Injektionsrate: 0.01

Abb. 3.13.: Maximale Verzögerung im Netz mit der Lastverteilung Transpose.

rein zufällige Lastverteilung selten einem anwendungsgetriebenen Kommunikationsmuster. Um diesen Umstand Rechnung zu tragen, wurden die drei synthetischen Muster Transpose, Bit-Reversal und Hot-Spot der Evaluierung hinzugefügt und unter den gleichen Bedingungen wie zuvor Random auch in den Simulationen eingesetzt.

Die Ergebnisse der Simulationen mit dem Sendemuster Transpose sind Abbildung 3.13 zusammengefasst. Ähnlich wie beim Einsatz des Sendemusters Random, sind bei Transpose und geringer Injektionsrate in Abbildung 3.13(a) deutliche Unterschiede bei der maximalen Verzögerung erkennbar. Die zusätzliche Belastung entspricht bei den Pfadlängen 2 und 4 etwa 48% (XY-Strategie), bzw. 16% (Staircase-Strategie). Die Verwendung der Staircase-Strategie sorgt auch bei diesem Sendemuster für kürzere Wartezeiten der Anwendungsnachrichten. Interessant ist auch, dass in diesem Muster neben der Simulation ohne Heartbeat-Nachrichten, die Staircase-Strategie ebenfalls nahezu konstante Wartezeiten bei den Anwendungsnachrichten bewirkt. Das bedeutet, dass mit der Platzierung der Kommunikationspaare durch Transpose bei jeder Kommunikation nicht mehr als etwa 7 Takte Wartezeit entstanden. Bei längeren Pfadlängen verhalten sich die Messwerte für die XY-Strategie wie erwartet und zeigen kürzere Wartezeiten bei Anwendungsnachrichten entlang der Außenseiten des Netzes.

Auch bei höheren Injektionsraten ist die Verwendung der Staircase-Strategie für Heartbeat-Nachrichten der XY-Strategie vorzuziehen. Bei einer Injektionsrate von 0.001 (Abbildung 3.13(b)) ist zu erkennen, dass die Referenz-Simulation nahezu identisch zu denen mit der Staircase-Strategie ist. Die Ergebnisse mit der XY-Strategie liegen jedoch weiterhin $\approx 10\%$ über der Referenz-Simulation. Erst am Sättigungspunkt ist ein Unterschied beider Routing-Strategien nicht mehr erkennbar. Die Kollisionen der Anwendungsnachrichten sind auch hier dominant und lassen die Heartbeat-Nachrichten an Relevanz verlieren.



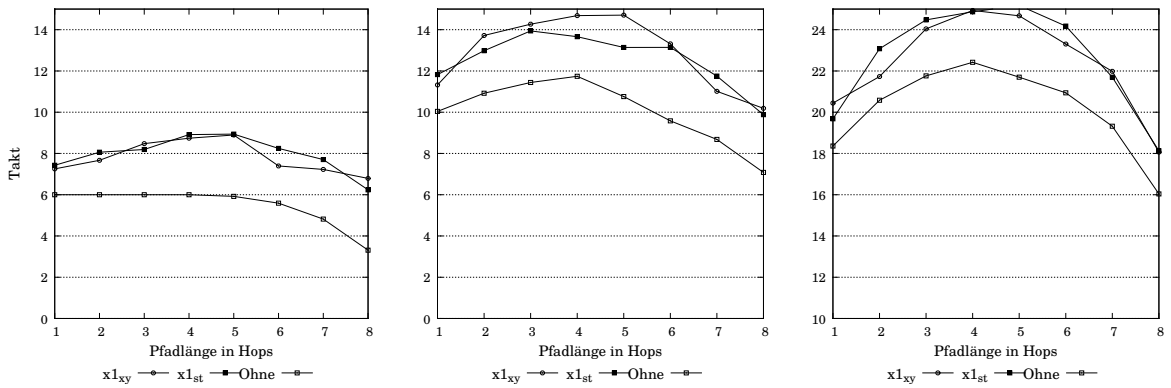
(a) Injektionsrate: 0.00001 (b) Injektionsrate: 0.001 (c) Injektionsrate: 0.01

Abb. 3.14.: Maximale Verzögerung im Netz mit der Lastverteilung Bit-Reversal.

Die Simulationsergebnisse des Sendemusters Bit-Reversal zeigen indes einen Fall, bei der die Staircase-Strategie leicht höhere Wartezeiten verursacht. Die resultierenden Wartezeiten sind in Abbildung 3.14 zusammengefasst. Die Verwendung der XY-Strategie bewirkt bei diesem Sendemuster insgesamt sehr ähnliche maximale Wartezeiten verglichen mit der Staircase-Strategie. Abgesehen von dem Ausreißer der Pfadlänge 3 in Abbildung 3.14(a) liegen die maximalen Wartezeiten der XY-Strategie etwa 5% unter denen der Staircase-Strategie. Dies liegt vor allem an der Art wie die Kommunikationspaare in diesem Sendemuster ausgewählt werden. Die Paarungen ergeben, dass der Großteil der Kommunikation das Zentrum des Netzes meidet und demnach hauptsächlich an den Rändern des Netzes stattfindet. Die Anwendungsnachrichten profitieren bei diesem Sendemuster wieder von der geringen Anzahl Heartbeat-Nachrichten im Randbereich des Netzes (vgl. Abbildung 3.5(a) und 3.6(a)).

Dazu kommt, dass beim Sendemuster Bit-Reversal etwa zwei Drittel der Kommunikation vertikal und damit orthogonal zu den Heartbeat-Nachrichten verläuft, die mit der XY-Strategie übertragen werden. Diese vertikale Ausrichtung ist das Resultat der Adress-Bit basierende Permutation des Bit-Reversal Verfahrens um die Kommunikationspaare zu bestimmen. Es ist ein gutes Beispiel dafür, welchen Einfluss ein gegebenes Sendermuster auf die tatsächliche Leistungsfähigkeit eines Netzes hat und zeigt zudem eine Schwäche der Staircase-Strategie auf. Wie auch schon im Abschnitt 3.2 angedeutet wurde, kann es auch mit der Verwendung der Staircase-Strategie für Heartbeat-Nachrichten unter gewissen Voraussetzungen zu längeren Verzögerungen der Anwendungsnachrichten kommen. Mit dem Bit-Reversal Sendemuster konnte so ein Fall identifiziert und konstruiert werden.

Dennoch bleibt festzustellen, dass die längeren Verzögerungen bei der Verwendung der Staircase-Strategie gering ausfallen und mit steigender Injektionsrate die Wartezeiten aller Simulationskonfigurationen erneut konvergieren. Bei diesem Sendemuster ist dies



(a) Injektionsrate: 0.00001 (b) Injektionsrate: 0.001 (c) Injektionsrate: 0.01

Abb. 3.15.: Maximale Verzögerung im Netz mit der Lastverteilung Hot-Spot.

bereits ab einer Injektionsrate von 0.001 erkennbar und somit früher als bei Random oder Transpose.

Zum Abschluss der Evaluierung der Staircase-Strategie, wurde die Simulation mit dem Hot-Spot Sendemuster konfiguriert und ausgeführt. Die Ergebnisse bezüglich der maximalen Verzögerung sind in der Abbildung 3.15 aufgetragen.

Das Hot-Spot Sendemuster für Anwendungsnachrichten stellt ein besonderes Szenario dar und ergibt mit den Heartbeat-Nachrichten ein etwas anderes Ergebnis. Während sich die bisherigen Werte der maximalen Verzögerungen mit und ohne Heartbeat-Nachrichten im Netz bei steigenden Injektionsraten immer stärker einander annäherten, ist dies hier nicht der Fall. Die Dauer der Verzögerungen der Anwendungsnachrichten durch Heartbeat-Nachrichten im Netz wächst stärker mit den steigenden Injektionsraten an, als bei den zuvor betrachteten Simulationen. Die Gründe dafür sind der Ort des jeweiligen Hot-Spots und die 60%*ige* Wahrscheinlichkeit, dass ein beliebiger Prozessorkern einen Hot-Spot als Ziel auswählt. Die Orte wurden so gewählt, dass sie genau auf den Achsen der FDU liegen. Anwendungsnachrichten sind also durch die XY-Strategie gezwungen, sich durch das Zentrum des Netzes zu bewegen. Da die Wahrscheinlichkeit dort sehr hoch ist, von einer Heartbeat-Nachricht blockiert zu werden, bleibt der Einfluss der Heartbeat-Nachrichten auf Anwendungsnachrichten auch bei hohen Injektionsraten erhalten.

Ferner zeigen sich keine signifikanten Unterschiede bei den Verzögerungen unter der Verwendung der unterschiedlichen Routing-Strategien für Heartbeat-Nachrichten. Dies ist bei diesem Testfall jedoch auch nicht erwartet worden, da die Anwendungsnachrichten bewusst durch das Zentrum des Netzes gleitet wurden und somit deren Verzögerungen für beide Routing-Strategien abschätzbar gleich ausfallen sollten.

Sendmuster	**ØDurchsatz**			**ØLatenz**			**max. Verzög.**		
	Ohne	R_{XY}	R_{ST}	Ohne	R_{XY}	R_{ST}	Ohne	R_{XY}	R_{ST}
Random	20%			12 Takte			0%	63,3%	38,3%
Transpose	68%			14 Takte			0%	46,7%	16,4%
Bit-Reversal	52%			12 Takte			0%	20,0%	50,6%
Hot-Spot	35%	40%		13 Takte			0%	21,3%	21,6%

Tab. 3.1.: Direkte Gegenüberstellung der Routing-Strategien XY- und Staircase-Routing.

3.3.6. Zusammenfassung der Evaluierung

Die in der Evaluierung angewendeten TDMA-Schemata ergibt, dass im Durchschnitt kein signifikanter Einfluss von den Heartbeat-Nachrichten auf Anwendungsnachrichten ausgeübt wird (siehe Tabelle 3.1). Der durchschnittliche Durchsatz und die Latenzzeiten der Anwendungsnachrichten sind auch in Gegenwart der Heartbeat-Nachrichten im Netz auf dem Niveau der Simulationen ohne Heartbeat-Nachrichten.

Eine genauere Untersuchung der Anwendungsnachrichten in Bezug auf deren maximalen Wartezeiten ergibt jedoch, dass diese durch die Verwendung der Staircase-Strategie gesenkt werden können. Die Tabelle 3.1 fast noch einmal alle relevanten Daten bzgl. der maximalen Wartezeiten zusammen. Bei den Sendemustern Random und Transpose sind die maximalen Verzögerungen deutlich geringer (25 – 30% Zeitersparnis). Bit-Reversal stellt sich hingegen als problematisches Sendemuster für Anwendungsnachrichten heraus. Hierbei werden unter der Verwendung der Staircase-Strategie etwa 30% längere maximale Wartezeiten gemessen. Nahezu identisch sind die maximalen Wartezeiten bei dem Sendemuster Hot-Spot. Hier beträgt der Unterschied zwischen beiden Routing-Strategien weniger als einem Prozent.

Die These aus Abschnitt 3.2 die besagt, dass eine gleichmäßigere Lastverteilung der Heartbeat-Nachrichten durch die Staircase-Strategie, einen positiven Effekt auf die Laufzeiten der Anwendungsnachrichten haben kann, konnte mit Hilfe der Untersuchung der maximalen Verzögerung untermauert werden. Es existieren Sendemuster wie das Bit-Reversal, bei der die maximalen Wartezeiten durch die Staircase-Strategie höher sind. Jedoch zeigten die Muster Random und Transpose deutlich geringere Verzögerungen.

4. Fehlerlokalisierung

Dieses Kapitel befasst sich mit einem Verfahren zur Lokalisierung von fehlerhaften Komponenten im Kommunikationsnetz unter Zuhilfenahme der bereits existierenden Heartbeat-Nachrichten. Das Verfahren besteht im Wesentlichen aus einem Approximationsverfahren und beinhaltet die folgenden vier Stufen:

(i) Vergleich der Heartbeat-Latenz gegenüber dem Erwartungswert und bei Abweichungen entsprechende Einstufung von beteiligten Netzkomponenten in *verdächtig*.

(ii) Mittels weiterer Heartbeat-Nachrichten fälschlich verdächtigte Komponenten per Ausschlussverfahren rehabilitieren.

(iii) Weitere Filterung der verdächtigten Komponenten durch Bestimmung lokaler Maxima.

(iv) Bei erfolgreicher Lokalisierung: Anpassung der eigenen Datenbasis bezüglich des aktuellen Zustands des Kommunikationsnetzes.

Um aus der Übertragungsdauer von Heartbeat-Nachrichten Informationen über den Zustand des Netzes zu gewinnen, greift das Lokalisierungsverfahren auf weiteres Wissen zurück. Die Zeitpunkte an denen die Heartbeat-Nachrichten abgesendet werden, sind im Vorfeld von der FDU im Rahmen des TDMA-Schema festgelegt worden und daher der FDU genau bekannt. Ferner ist durch die deterministische Natur der Staircase-Strategie der Pfad einer Nachricht berechenbar. Zusätzlich sorgt das QoS zusammen mit dem TDMA-Schema dafür, dass Heartbeat-Nachrichten weder untereinander noch mit Anwendungsnachrichten kollidieren. Auf diese Weise entsteht ein Determinismus, der es erlaubt zu berechnen, wann und wo im Netz eine Heartbeat-Nachricht existiert, beziehungsweise wann diese erwartungsgemäß bei der FDU eintrifft. Abweichungen von der erwarteten Latenz sind also ein Indiz dafür, dass die Nachricht nicht planmäßig durch das Netz übertragen wurde.

Im Folgenden wird zunächst anhand eines Beispiel-Szenarios das Grundprinzip der Fehlerlokalisierung verdeutlicht. Anschließend werden alle notwendigen Anpassungen am *Heartbeat-Sendeverhalten*, der *Kerngruppierung* und der *Routing-Strategie* beschrieben. Diese Anpassungen sind nötig um das Lokalisierungsverfahren vollständig und flächendeckend umsetzen zu können. Bei den Anpassungen wurde darauf geachtet, nur minimale Änderungen am Sendeverhalten, des Gruppierungsmechanismus und der Routing-Strategie vorzunehmen, um das Grundsystem der FDU unverändert zu lassen. Das Ziel war

es, beide Mechanismen (sowohl die native FDU-Kernüberwachung als auch die Fehlerlokalisierung im Netz) parallel betreiben zu können. Anschließend werden die zusätzlichen Kosten, die aus den Anpassungen hervorgehen, beschrieben. Als weiterführende Arbeit schließt das Kapitel mit der Untersuchung von verschiedenen Typen multipler Fehler und toroidalen Netztopologien ab.

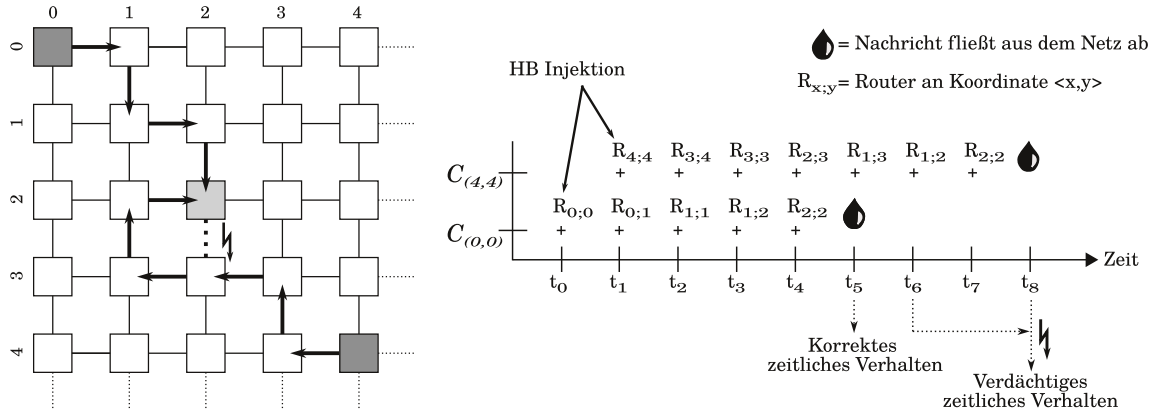
4.1. Beispielszenario

Zu Beginn dieses Beispielszenarios werden zunächst die Rahmenbedingungen skizziert und anschließend das Lokalisierungsverfahren anhand des Beispielszenarios erläutert. Abbildung 4.1(a) zeigt in Anlehnung an Kapitel 2.3 ein Teil des zugrunde liegenden Kommunikationsnetzes der angepassten TERAFLUX-Architektur. Aus Gründen der Übersicht wurden die einzelnen Router und deren angeschlossene Prozessorkerne in den quadratischen Kacheln zusammengefasst. Die Verbindungsleitungen zwischen den Routern sind als Linien dargestellt. Es ist zu beachten, dass in diesem Beispielszenario die Verbindungen als einzelne bidirektionale Leitung angenommen werden, auf denen nur in jeweils eine Richtung gesendet werden kann (*half duplex*) und im Folgenden als $R_{(x,y)} \leftrightarrow R_{(x\prime,y\prime)}$ abgekürzt.

Um die Orientierung zu erleichtern sowie zur späteren Referenznahme wurde ein Koordinatensystem um den Prozessoraufbau gelegt. Im Ursprung des Koordinatensystems befindet sich der Kern $C_{(0,0)}$ und ist einer der beiden Akteure in diesem Beispielszenario. $C_{(4,4)}$ ist der zweite Akteur und befindet sich am unteren rechten Rand des Prozessors. Im Zentrum der Kerngruppe befindet sich die FDU (graues Quadrat), die eingehende Heartbeat-Nachrichten von den umliegenden Prozessorkernen empfängt und auswertet. Die beiden Prozessorkerne $C_{(0,0)}$ und $C_{(4,4)}$ senden gemäß des TDMA-Schemas ihre Heartbeat-Nachrichten an die FDU. Alle Prozessoren in dieser Abbildung stellen einen FDU-überwachten Cluster dar.

Die jeweiligen Pfade, welche die Heartbeat-Nachrichten im Netz zurücklegen, wurden durch die Routing-Strategie Staircase erzeugt und sind durch Pfeile zwischen den Routern hervorgehoben. Eine Besonderheit stellt die gestrichelte Linie $R_{(2,2)} \leftrightarrow R_{(2,3)}$ dar. Hierbei handelt es sich um eine fehlerhafte Verbindungsleitung zwischen den Routern $R_{(2,2)}$ und $R_{(2,3)}$. Sie kann damit nicht für den Transport von Nachrichten verwendet werden. Das bedeutet, dass sowohl Anwendungsnachrichten als auch Heartbeat-Nachrichten eine alternative Route verwenden müssen, sofern $R_{(2,2)} \leftrightarrow R_{(2,3)}$ Teil des ursprünglichen Nachrichtenpfades ist. In diesem Beispiel wählt der Router $R_{(2,3)}$ die westliche Richtung um die Heartbeat-Nachrichten zur FDU umzuleiten. Dies führt zu einem Umweg der Heartbeat-Nachricht von $C_{(4,4)}$ und gleichzeitig zu einem verspäteten Eintreffen bei der FDU[1].

[1] In diesem Netzaufbau können Fehler an bestimmten Positionen auftreten, deren Umleitung nicht zwangsläufig zu einer Verzögerung der Ankunftszeit führen. Auf diesen Fall wird im späteren Verlauf



(a) Routen durch das Netz zweier Heartbeat-Nachrichten. Die Nachricht von $C_{(4,4)}$ wurde an $R_{(2,3)} \leftrightarrow R_{(2,2)}$

(b) Zeitliches Verhalten im Fehlerfall

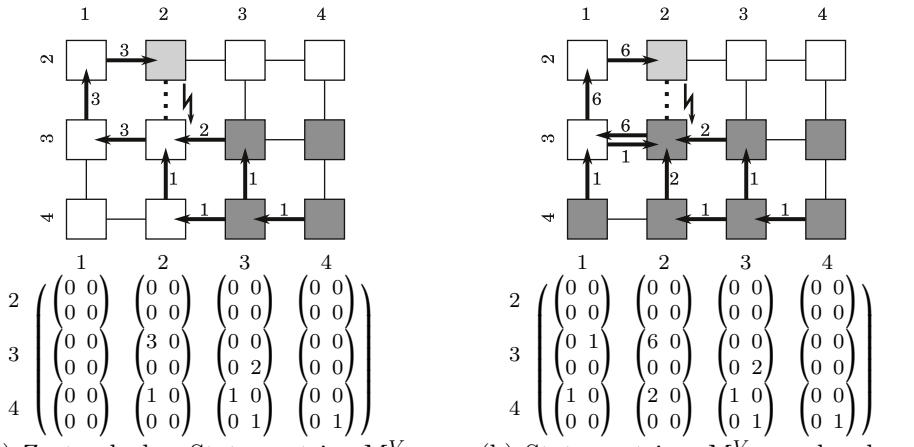
Abb. 4.1.: Beispielszenario eines fehlerhaften Netzes

Der Zeitstrahl auf der Abbildung 4.1(b) bildet das beschriebene Beispielszenario noch einmal aus der zeitlichen Sicht ab. Der Kern $C_{(0,0)}$ versendet seine Nachricht zum Zeitpunkt t_0. $C_{(4,4)}$ setzt seine Heartbeat-Nachricht zum Zeitpunkt t_1 ab. Die Heartbeat-Nachricht von $C_{(0,0)}$ trifft ohne Umwege auf dem ihr vorgeschriebenem Pfad bei der FDU ein. Die Nachricht fließt planmäßig zum Zeitpunkt t_5 aus dem Netz an der FDU ab. Die Heartbeat-Nachricht von $C_{(4,4)}$ hingegen trifft bei der Übermittlung auf die fehlerhafte Verbindungsleitung $R_{(2,2)} \leftrightarrow R_{(2,3)}$ und wird wie oben beschrieben umgeleitet. Dadurch verschiebt sich die Ankunft vom Zeitpunkt t_6 zum späteren Zeitpunkt t_8.

4.2. Die Fehlerquelle eingrenzen

Diese erste Abweichung in der Übertragungsdauer ist für die FDU das Indiz, dass die Heartbeat-Nachricht von $C_{(4,4)}$ an einer bestimmten Position im Netz anders als geplant weitergeleitet werden musste. Die genaue Position lässt sich jedoch nicht aus der zeitlichen Verzögerung einer einzelnen Heartbeat-Nachricht ermitteln. Daher markiert die FDU den gesamten ursprünglich vorgesehenen Pfad dieser Heartbeat-Nachricht als *verdächtig*. Die Markierung findet in einer von zwei Zustandsmatrizen statt, welche die FDU bereitstellt. Die erste Matrix M^V speichert für jede Verbindungsleitungen die Anzahl an verspäteten Heartbeat-Nachrichten, die auf ihr übertragen wurden. Die zweite Matrix M^F dient zum Speichern bereits lokalisierter Fehler.

dieses Kapitel weiter eingegangen. An dieser Stelle wird jedoch aus Vereinfachungsgründen nicht weiter auf diese Fehlerorte eingegangen.



(a) Zustand der Statusmatrix M^V nach dem Erhalt der ersten Verzögerten Heartbeat-Nachricht

(b) Statusmatrix M^V nach dem Erhalt von drei verspäteten Heartbeat-Nachrichten

Abb. 4.2.: Routing-Verhalten bei Heartbeat-Nachrichten im Fehlerfall. Die dunkel grauen Quadrate symbolisieren die Kerne deren Heartbeat-Nachricht von dem Fehler betroffen ist. Die Zahlen an den Pfeilen geben die jeweilige Anzahl von verspäteten Heartbeat-Nachrichten an, die auf ihnen weitergeleitet wurden.

Die Werte der Matrix M^V werden durch zwei Ereignisse beeinflusst. Ist eine Heartbeat-Nachricht verspätet zur FDU übermittelt worden, werden alle Komponenten, die gemäß der Routing-Strategie am Transport dieser Nachricht beteiligt waren, um den Wert 1 inkrementiert. Mit anderen Worten, diese Komponenten werden *vorläufig verdächtigt* fehlerhaft zu sein. Im gegensätzlichen Fall, also wenn die Heartbeat-Nachricht zum erwarteten Zeitpunkt eintraf, werden die entsprechenden Einträge in M^V um den Wert 1 dekrementiert. Das Dekrementieren wird fortan als *Rehabilitierung* bezeichnet, da eine zuvor verdächtige Komponente nun seine funktionale Korrektheit nachgewiesen hat.

Abbildung 4.2 ist ein Ausschnitt aus dem obigen Beispielszenario und zeigt zwei Momentaufnahmen in denen illustriert wird, wie die Heartbeat-Nachrichten umgeleitet werden und wie sich der Zustand der Matrix M^V nach dem Erhalt dieser Nachrichten ändert. Im oberen Teil der Abbildungen sind die sendenden Prozessorkerne als dunkelgraue Quadrate gekennzeichnet. Die jeweiligen Pfade der verspäteten Heartbeat-Nachrichten sind als Pfeile mit Werten aufgetragen, wobei die Werte eines Pfeils beschreiben, wieviele Nachrichten über diese Verbindungsleitung übertragen wurden. Der untere Teil der Abbildungen 4.2(a) und 4.2(b) zeigt den Inhalt der Statusmatrix M^V. Die Untermatrizen $M^V_{x,y}$ innerhalb M^V dienen der besseren Übersicht und folgen dem Aufbau $M^V_{x,y} = \binom{NO}{SW}$. Sie enthalten die ausgehenden Verbindungsleitungen des Routers $R_{(x,y)}$. Die Zeichen N, O, S und W definieren die entsprechende Himmelsrichtung in die eine Verbindungsleitung "zeigt". Werte > 0 in $M^V_{x,y}$ weisen auf eine verdächtige Komponente hin.

Um die Lokalisierung zu illustrieren, wird in Abbildung 4.2(a) das anfängliche Szenario zunächst um die Prozessorkerne $C_{(3,3)}$, $C_{(3,4)}$ und $C_{(4,3)}$ erweitert. Jeder dieser Ker-

ne (dunkelgraue Quadrate) sendet jeweils eine Heartbeat-Nachricht an die FDU. Die daraus resultierenden Pfade sind als Pfeile hervorgehoben. Die von den Kernen $C_{(3,3)}$ und $C_{(4,3)}$ stammenden Nachrichten wurden von $R_{(2,3)}$ umgeleitet, was gleichzeitig eine erhöhte Übertragungsdauer zu Folge hat und somit den Wert an den jeweiligen Pfeilen erhöht. Die Heartbeat-Nachricht von $C_{(4,3)}$ hingegen wurde auf einem Pfad durch das Netz geleitet, der nicht die Verbindung $R_{(2,2)} \leftrightarrow R_{(2,3)}$ verwendet und konnte daher die FDU planmäßig erreichen. Die Ankunftszeiten der einzelnen Heartbeat-Nachrichten liefern der FDU weitere Indizien über fehlerhafte Komponenten im Netz. Ähnlich wie die Nachricht von $C_{(4,4)}$ zeigen die Nachrichten von $C_{(3,3)}$ und $C_{(4,3)}$ eine Abweichung zwischen der erwarteten und der tatsächlichen Ankunftszeit. Entsprechend werden die jeweiligen Einträge der verwendeten Komponenten in M^V aktualisiert. Wie aus der Matrix in Abbildung 4.2(a) hervorgeht, stieg der Wert des Eintrags $M^V_{2,3}$ für die nördliche Verbindungsleitung im Vergleich zu dessen benachbarten Komponenten stark an.

Das Ausschlussverfahren wird deutlich, wenn das Szenario weiter geführt wird, bis alle Kerne eine Heartbeat-Nachricht an die FDU geschickt haben. Vormals verdächtigte Komponenten konnten mit Hilfe von anderen Heartbeat-Nachrichten, die keine verzögerte Ankunft hatten, rehabilitiert werden. Aus den restlichen Indizien entsteht die vorerst finale Version der Zustandsmatrix M^V, dargestellt in Abbildung 4.2(b).

Es fällt auf, dass noch immer einige Komponenten im Netz als verdächtig einstuft sind, obwohl diese korrekt funktionieren. Der letzte Schritt der Lokalisierung besteht folglich daraus, aus den verbliebenen Komponenten diejenigen auszuwählen, die einen Höchstwert in M^V haben. Hierbei ist jedoch zu beachten, dass nicht etwa das globale Maximum gesucht wird, sondern alle lokalen Maxima. Denn im Fall von multiplen Fehlern im Netz würde die Suche nach einem globalen Maximum möglicherweise nur ein fehlerhaftes Element hervorheben. Die Suche nach lokalen Maxima hingegen kann auch mehrere Fehler isolieren und lokalisieren.

4.3. Bestimmung der Fehlerquelle

Die FDU wurde zur genauen Bestimmung der fehlerhaften Komponente um den Filter $\mathcal{F}$ erweitert, der die lokalen Maxima innerhalb M^V bestimmt. Der Filter besteht im wesentlichem aus einem Faltungsoperator, der auf jedes Element $M^V_{x,y}(d)$ angewendet wird und somit den Eintrag in $M^F_{x,y}(d)$ bestimmt. $\mathcal{F}$ ist definiert durch

$$M^F_{(x,y)}(d) = \begin{cases} 1, & \text{wenn } \mathcal{F}(x,y,d) > 0 \\ 0, & \text{sonst} \end{cases}$$

mit

$$d \in D = \{N, O, S, W\}$$

und

$$\mathcal{F}(x,y,d) \coloneqq M^V_{(x,y)}(d) - max(M^V_{(x-1,y)}(D), M^V_{(x+1,y)}(D), M^V_{(x,y-1)}(D), M^V_{(x,y+1)}(D))$$

Die $max()$-Funktion bestimmt hierbei den höchsten Wert aller Verbindungsleitungen der direkten Nachbar-Router, mit $x \pm 1$ der horizontalen Nachbarn und $y \pm 1$ der vertikalen Nachbarn. Ist die Differenz zwischen dem lokalen Wert der Verbindungsleitung d und dem ermittelten Maximalwert aller Nachbarn > 0 liegt ein lokales Maxima vor.

Wurde ein lokales Maximum gefunden, wird in M^F an der entsprechenden $< x, y, d >$-Koordinate die fehlerhafte Komponente mit dem Wert 1 markiert. Da sich durch die Markierung der Verbindungsleitungen auch ein Totalausfall des Routers kodieren lässt, indem alle Verbindungsleitungen als fehlerhaft markiert werden, kann auf einen gesonderten Wert bzgl. des Routers in den Matrizen verzichtet werden.

Nach der erfolgreichen Lokalisierung der Fehler wird abschließend die FDU-eigene Datenbasis so angepasst, dass die nun bekannten Fehler keine Verzerrung auf die weiteren Messungen der FDU haben. Hierzu werden die Werte der erwarteten Ankunftszeiten der betroffenen Kerne entsprechend ihrer neuen Route durch das Netz angepasst und in das TDMA-Schema eingebracht. Nachdem alle lokalen Maxima in M^F eingetragen wurden und die Anpassung der Datenbasis durchgeführt wurde, wird M^V vollständig auf 0 zurückgesetzt und die nächste Überwachungsrunde beginnt.

Die anschließende Evaluierung dieses Lokalisierungsverfahrens wird zeigen, dass auch multiple Fehler im Netz existieren dürfen, ohne die Lokalisierung zu beeinträchtigen, solang diese nicht in speziellen Mustern auftreten. Zuvor wird jedoch das Kommunikationsnetz so erweitert, dass es dem de facto Standard entspricht. Dazu gehört das Ersetzen der in diesem Beispiel verwendeten bidirektionalen Verbindungsleitungen durch paarweise unidirektionale Verbindungsleitungen, die einen simultanen Datentransfer in beide Richtungen erlauben. Diese Änderungen am Netz erfordern jedoch für das Lokalisierungsverfahren spezielle Anpassungen bezüglich des *Heartbeat-Sendeverhaltens*, der *Kerngruppierung* zu einem FDU-überwachten Cluster und der *Routing-Strategie*, welche in den folgenden Abschnitten diskutiert werden.

4.4. Notwendige Anpassungen am Heartbeat-Mechanismus

Wie im obigen Fall-Beispiel bereits erwähnt, existieren Situationen bei denen es nicht zwangsläufig zu einer erhöhten Übertragungsdauer kommt. Tatsächlich sorgt das Prinzip des Minimal-Path-Routing (MPR) (vgl. Abschnitt 2.1.4) dafür, dass nur Fehler auf den Verbindungsleitungen der *FDU-Achsen* auf natürliche Weise die Übertragungsdauer erhöht. Bei Fehlern abseits dieser Achsen steht der Staircase-Strategie eine alternative

Verbindung zur Verfügung, die keinen Einfluss auf die ursprünglich errechnete Übertragungsdauer einer Nachricht hat. Um dennoch eine Verzögerung der Ankunft zu erreichen, muss eine künstliche Verzögerung erzeugt werden, indem die Nachricht auf einer Verbindungsleitung weitergeleitet wird, die den Abstand zum Ziel vergrößert, statt verkürzt. Der daraus entstandene Umweg erzeugt dann die notwendige Verzögerung, verletzt jedoch gleichzeitig das MPR-Prinzip der Staircase-Strategie, was eine Anpassung dieser Strategie nötig macht.

Wie bereits im Abschnitt 2.1.4 beschrieben wurde, besitzt die Staircase-Strategie durch die Auswahl einer alternativen Route die Fähigkeit, fehlerhafte Verbindungsleitungen zu meiden. Während die Auswahlregeln darauf ausgelegt sind, dass MPR-Prinzip einzuhalten und Deadlocks zu vermeiden, sind diese Regeln speziell für Heartbeat-Nachrichten verändert worden, sodass die favorisierte Verbindungsleitung eine von denen ist, die den Weg zur FDU verlängert. Hier ist lediglich die Reihenfolge, in der die alternativen Verbindungsleitungen von der Routing-Strategie vorgeschlagen werden, geändert worden. Somit wird im Fehlerfall zunächst immer versucht eine wegeverlängernde Leitung als Alternative zu verwenden.

Neben den Fehlern die zunächst keinen Einfluss auf die Übertragungsdauer einer Heartbeat-Nachricht haben, existieren Verbindungsleitungen die mit dem bisher vorgestellten Prinzip der Fehlerlokalisierung nicht überwacht werden können. Bei diesen *toten Winkeln* handelt es sich um Verbindungsleitungen auf denen entweder keine Heartbeat-Nachrichten weitergeleitet werden oder es keine alternative Verbindungsleitung gibt, die einen Einfluss auf die Übertragungsdauer einer Heartbeat-Nachricht hat. Diese toten Winkel lassen sich in vier Teilbereiche des Netzes eingliedern:

(i) Am Rand, innerhalb eines Clusters.

(ii) Am Rand, außerhalb eines Clusters.

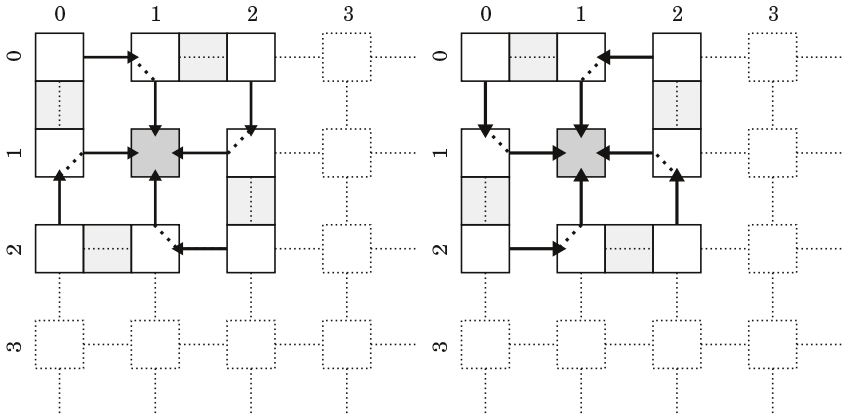
(iii) Verbindungsleitungen, die nicht in Richtung FDU “zeigen”.

(iv) In den Ecken des Prozessors.

Die Teilbereiche (i), (ii) und (iii) beziehen sich auf Verbindungsleitungen, die nicht von Heartbeat-Nachrichten verwendet werden. Punkt (iv) hingegen ist ein Bereich in dem keine alternative Verbindungsleitung existiert, welche die Übertragungsdauer erhöht. Im Folgenden werden die Punkte (i) bis (iv) separat betrachtet und die notwendigen Erweiterungen an dem Heartbeat-Mechanismus erläutert, um die jeweiligen toten Winkel zu kompensieren.

4.4.1. Am Rand innerhalb der Cluster

An jedem der vier Ränder einer Clusters existieren Bereiche, die nicht von Heartbeat-Nachrichten verwendet werden. In Abbildung 4.3 sind diese Bereiche mit grauen Quadraten markiert. Um die toten Winkel zu entfernen, wurde die Staircase-Strategie gespiegelt



(a) Standard Staircase Routing (b) Alternierte Staircase Routing

Abb. 4.3.: Behandlung der toten Winkel durch den Einsatz von alternierendem Routing

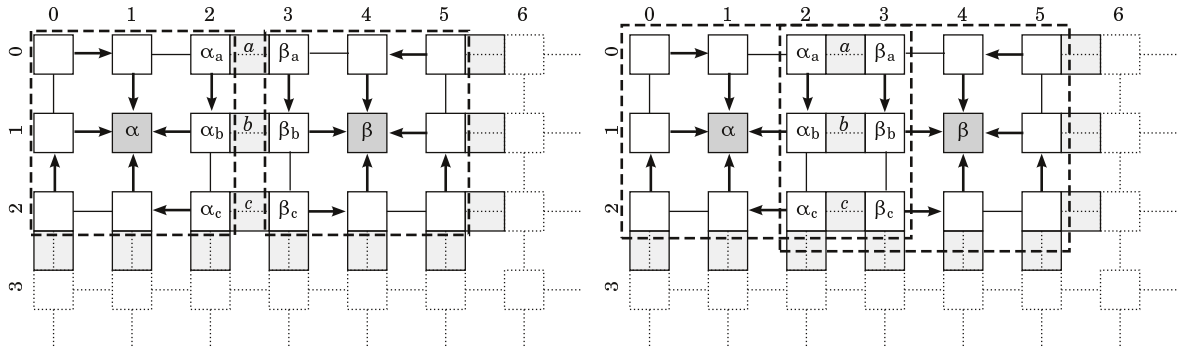
(vgl. Abbildung 4.3(b)), sodass sich der Fluss der Nachrichten gewissermaßen gegen den Uhrzeigersinn bewegt. Die toten Winkel an den Rändern werden ebenfalls gespiegelt, sodass sie sich jetzt an Stellen befinden, wo zuvor noch Heartbeat-Nachrichten entlang geleitet wurden. Alterniert man zwischen den beiden Routing-Strategien von einer Heartbeat-Nachricht zur nächsten, werden diese toten Winkel aufgelöst.

Damit ein Router erkennt, welche Routing-Strategie für die aktuelle Heartbeat-Nachricht anzuwenden ist, wurde im Flit-Kopf dieser Nachrichten ein Flag hinzugefügt. Dieses signalisiert dem Router, welche Routing-Strategie verwendet werden soll. Das Routing-Flag im Flit-Kopf wird ausschließlich vom sendenden Prozessorkern gesetzt. Hierbei wird das Flag immer dann gesetzt, wenn in der zuvor gesendeten Heartbeat-Nachricht das Flag nicht gesetzt wurde.

Bei diesem Verfahren ist zu beachten, dass die Lokalisierung eines Fehlers in den toten Winkel mehr Zeit benötigt, da die Nachrichten durch das Alternieren zwischen den Routing-Strategien nur in jeder zweiten Runde die entsprechenden Verbindungsleitungen nutzen.

4.4.2. Am Rand außerhalb des Clusters

In der ursprünglichen Version der FDU-basierenden Fehlererkennung sind alle Cluster rechteckig und disjunkt voneinander. Daher gibt es keine Heartbeat-Nachricht, die über die virtuellen Grenzen eines Clusters hinweg durch das Netz transportiert wird. Dies sorgt jedoch dafür, dass die Bereiche zwischen dem jeweiligen Cluster nicht überwacht werden. In Abbildung 4.4(a) wurde dieser tote Winkel durch graue Quadrate hervorgehoben. Um beispielsweise die toten Winkel der Verbindungsleitungen a, b und c zu entfernen, existieren zwei mögliche Herangehensweisen:



(a) Tote Winkel zwischen zwei Kerngruppen

(b) Entfernen der toten Winkel durch überlappende Kerngruppen

Abb. 4.4.: Behandlung der toten Winkel durch den Einsatz von alternierendem Routing

(i) Das Verschieben des Clusters von FDU β um eine Koordinate nach links, ohne Anpassung der Clustergröße.

(ii) Die Vergrößerung eines oder beider Cluster um die FDUs α und β.

Auf diese Weise überlappen sich beide Cluster, sodass, wie in Abbildung 4.4(b) zu erkennen ist, die Kerne α_a, α_b, α_c, β_a, β_b und β_c jeweils beiden Clustern angehören.

Welche der beiden oben genannten Herangehensweisen zum Erreichen der Überlappung eingesetzt werden, hängt von der aktuellen Clustergröße und deren Lage auf dem Prozessor ab. Aus dem vorhergehenden Kapitel über die Kosten für Heartbeat-Nachrichten ist bekannt, dass die angestrebte Clustergröße beispielsweise ein 5×5 Quadrat ist. Dies ist jedoch ein Richtwert mit oberen und unteren Grenzwerten. Die tatsächliche Größe kann also aufgrund von Umstrukturierungen variieren. Ähnliches gilt für das Verschieben von Clustern. Während Kerngruppen im Zentrum des Prozessors gewissermaßen frei verschoben werden können, sind Verschiebungen am Rand des Prozessors nur in bestimmte Richtungen möglich. So könnte beispielsweise der Cluster um β aus Abbildung 4.4(a) sowohl nach links und rechts als auch nach unten verschoben werden. Die Kerngruppe um α lässt sich hingegen nur nach rechts und unten verschieben.

4.4.3. Entgegen der Senderichtung

Anders als im vereinfachten Fall des Beispielszenarios, bei dem bidirektionale Verbindungsleitungen angenommen wurden, um Router untereinander zu verbinden, sind paarweise unabhängige unidirektionale Verbindungen der Stand der Technik. Unidirektionale Verbindungsleitungen bringen jedoch neue Komplexität in das Netz, wenn es darum geht, fehlerhafte Komponenten durch ein rein nachrichtenbasierendes Verfahren zu lokalisieren. Das Hauptproblem hierbei ist, dass die nun zusätzlich verfügbaren Verbindungsleitungen - wie auch schon bei den Randbereichen - nicht zwingend von Heartbeat-

Nachrichten verwendet werden. Teilt man die Leitungsrichtungen vom Prozessorkern zur FDU in die zwei Kategorien *wegverkürzend* und *wegverlängernd* ein, dann sind die wegverlängernden Leitungen jene, die nicht von Heartbeat-Nachrichten verwendet werden und entsprechen dem zur Folge einem toten Winkel. Abbildung 4.5 verdeutlicht dieses Problem. Die durchgezogenen Linien stellen die von Heartbeat-Nachrichten verwendeten Verbindungsleitungen auf den Pfaden zur FDU dar. Gestrichelte Linien repräsentieren die Gegenrichtung und gleichzeitig den toten Winkel. Um diese toten Winkel zu entfernen muss das Lokalisierungsverfahren also erweitert werden, sodass auch diese Verbindungsleitungen verwendet werden. Nachfolgend werden die unidirektionalen Verbindungsleitungen mit $R_{(x,y)} \overset{D}{\rightarrow} R_{(x\prime,y\prime)}$ abgekürzt dargestellt.

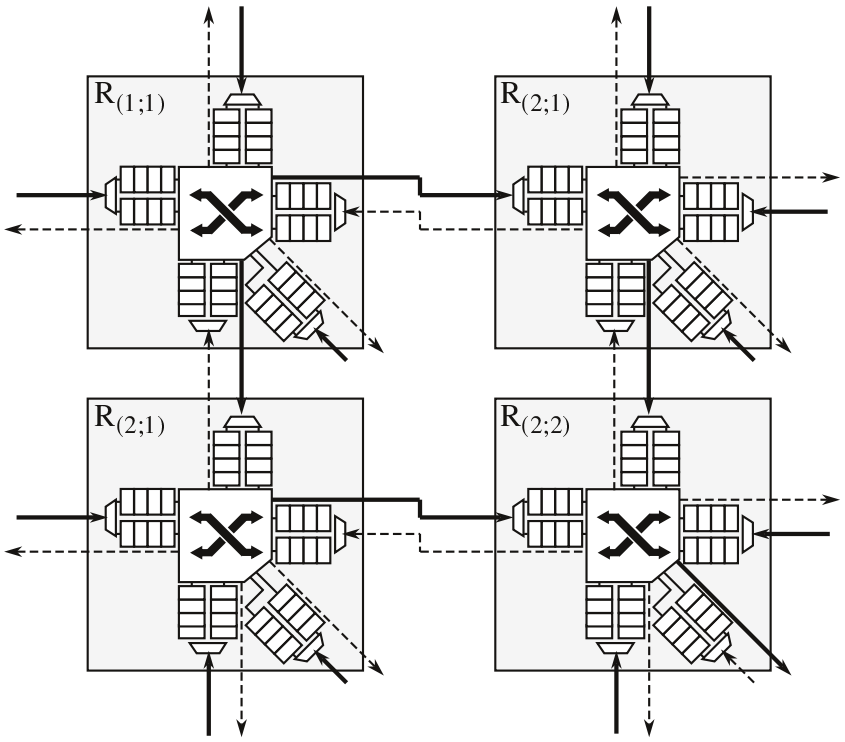


Abb. 4.5.: Paarweise unidirektionale Verbindungsleitung:
Die gestrichelten Linien zeigen die Leitungen, die von der FDU (bei $R_{(2,2)}$) weg "zeigen". Sie werden nicht von Heartbeat-Nachrichten genutzt und stellen somit einen weiteren toten Winkel da.

Bisher war das Verhalten der Prozessorkerne so, dass sie gemäß dem von der FDU definiertem TDMA-Schema Heartbeat-Nachrichten sendeten. Die Prozessorkerne senden also selbstständig Heartbeat-Nachrichten zur FDU, was als *push*-Protokoll bezeichnet wird. Im Gegensatz dazu existiert das *poll*-Protokoll. Bei diesem Protokoll wird zunächst eine Anfrage (*request*) an einen Kommunikationspartner gesendet, der die Anfrage beantwortet (*response*). Man spricht dabei auch von einer synchronen Kommunikation. Übertragen auf Heartbeat-Nachrichten bedeutet dies, dass beim Einsatz des poll-Protokolls die Prozessorkerne auf eine Anfrage der FDU warten, und erst dann eine

Heartbeat-Nachricht als Antwort schickten. Da auch die Anfragen der FDU eine Art der Heartbeat-Nachricht ist und mit den Staircase-Strategien durch das Netz übermittelt werden, werden nun in beide Richtungen Heartbeat-Nachrichten gesendet und somit auch die entsprechenden Verbindungsleitungen verwendet. Zusätzlich kommt die Erweiterung aus den vorhergehenden Unterkapiteln zum Einsatz. Das alternierende Routing sorgt nicht nur dafür, die toten Winkel an den Gruppenrändern zu beseitigen, sondern auch, dass alle wegverlängernden von den Anfragen verwendet werden.

Für die eigentliche Lokalisierung von Fehlern bedeutet diese Erweiterung, dass eine Antwort nun aus zwei unterschiedlichen Teilpfaden besteht; dem request-Pfad und dem response-Pfad. Das führt dazu, dass die FDU zunächst nicht unterscheiden kann, ob die request-Nachricht oder die response-Nachricht verzögert wurde. Somit muss die FDU zunächst beide Teilpfade als mögliche Fehlerquelle betrachten. Dies stellt insofern kein Problem dar, da die Nachrichten von benachbarten Prozessorkernen wieder für die Rehabilitation der funktionierenden Komponenten sorgen.

4.4.4. Eckbereiche des Prozessors

Die Eckbereiche des Prozessors stellen ein gesondertes Problem dar. Es lässt sich nicht mit Anpassungen der Routing-Strategie oder dem Sendeverhalten (push und poll) lösen. Das Problem ist die Anzahl der ausgehenden Verbindungen eines Eck-Routers. Alle anderen Router im Netz verfügen über mindestens drei Verbindungsleitungen, von denen wenigstens eine als wegverlängernd eingesetzt werden kann. Bei den Eck-Routern ist die Zahl der Ausgangsleitung in einem 2D-Gitter jedoch auf zwei begrenzt und keine dieser beiden Verbindungen vergrößert den Abstand zur FDU. Das bedeutet also, dass eine fehlerhafte Verbindungsleitung nicht mit den bisher vorgestellten Mitteln der Nachrichten basierende Fehlerlokalisierung ermittelt werden kann. Um Fehler dieser Verbindungsleitung dennoch zu erkennen, müssen Änderungen am Design des Netzes oder des Routers vorgenommen werden.

4.4.4.1. Verzögerung durch den Router

Betrachtet man das Router-Design aus Kapitel 2.1.2 lassen sich Ressourcen identifizieren, welche bei Router an den Ecken des Prozessors gewissermaßen nicht genutzt werden. An den Ecken und am Rand des Prozessors haben die Router bedingt durch fehlende Nachbar-Router weniger Ein- und Ausgangsleitungen. Dort sind es zwei Leitungspaare an den Ecken und drei Leitungspaare am Rand des Prozessors. Durch das Fehlen der dazugehörigen Puffer und geringen Komplexität der Kreuzschiene, verbrauchen diese Router weniger Platz, als ein Router im Inneren des Prozessors, welcher über die vollen vier Leitungspaare verfügt. Voraussetzung für dieses Konzept ist jedoch, dass die Eck-

und Rand-Router nicht noch zusätzliche Geräte (z.B. Ein-/Ausgabegeräte oder externer Speicher) angeschlossen haben.

Die Idee hier ist, den geringeren Platzverbrauch eines Eck-Routers auszunutzen in dem ein weiterer Puffer-Speicher implementiert wird. Dieser Puffer dient als eine Art zusätzliche Pipeline-Stufe des Routers. Müsste eine Heartbeat-Nachricht durch eine fehlerhafte Verbindungsleitung alternativ geroutet werden, sorgt die Routing-Logik dafür, dass eine ausgehende Heartbeat-Nachricht zunächst nicht die Ausgangsleitung verwendet, sondern stattdessen in diesem Puffer-Speicher zwischengespeichert wird. Erst im nächsten Takt[2] wird dann die Nachricht aus dem Puffer heraus auf die Ausgangsleitung gelegt und versendet. Auf diese Weise wird eine künstliche Verzögerung der Heartbeat-Nachricht erzeugt und die FDU kann auch diese fehlerhaften Verbindungen lokalisieren.

4.5. Kosten durch die Erweiterungen

Die oben beschriebenen Erweiterungen erzeugen ein zusätzliches Datenvolumen, welches über das Netz abgewickelt werden muss. In diesem Abschnitt des Kapitels werden die zusätzlichen Kosten aufgeführt, welche durch die Erweiterungen des Routings, des Heartbeat Sendeverhaltens und der Kern-Gruppierungen entstehen.

Als Grundlage dieser Berechnung dienen die Cluster, die jede FDU um sich herum aufspannt. Die dazugehörigen Prozessor-Kerne $c_{x,y}$ werden dazu in der Menge C zusammengefasst. Zusätzlich wird die Konstante LB als Verbindungsbreite in Bit definiert, welche die *Bandbreite pro Takt* zwischen zwei benachbarten Routern festlegt.

Des Weiteren ist für verschiedene Berechnungen auch die Wegstrecke, die eine Heartbeat-Nachricht zurücklegt, ein wesentlicher Faktor. Denn je länger(/kürzer) der Weg einer Nachricht durch das Netz, desto größer(/kleiner) ist die zu bewältigende Belastung im Netz. Zwar belegt eine Heartbeat-Nachricht in einem einzelnen Zeitschritt immer nur eine Ressource[3], jedoch muss das Netz die Nachricht über verschiedene Distanzen hinweg transportieren. Diese Distanzen lassen sich für Heartbeat-Nachrichten einfach durch die Bestimmung der Manhattan-Distanz

$$MD(c, fdu) = |x_c - x_{fdu}| + |y_c - y_{fdu}|$$

errechnen.

[2] Abhängig von der Art der Implementierung können auch mehrere Takte gewartet werden, bis die Nachricht gesendet wird.

[3] Es gilt weiterhin, eine Heartbeat-Nachricht besteht nur aus einem Flit-Kopf.

4.5.1. Alternierendes Routing

Das alternierende Routing besteht aus zwei Erweiterungen die zusätzliche Kosten verursachen. Der Flit-Kopf, in dem unter anderem die Routing-Informationen der Nachricht gespeichert sind, musste erweitert werden. Diese Erweiterung beschränkt sich auf ein einzelnes Bit, welches das Datenvolumen einer Nachricht erhöht. Dies soll anhand aktueller Netzentwürfe verdeutlicht werden. Intels Forschungsprozessor *SCC* verfügt über 128Bit breite Verbindungsleitungen zwischen den Routern. Würde der Flit-Kopf einer Heartbeat-Nachricht innerhalb dieses Netzes verschickt, entspricht der zusätzliche Overhead durch das Bit im Header $\frac{1}{128} \approx 0,8\%$. Nimmt man zum Vergleich den *Tile64*, der über 32Bit breite Verbindungsleitungen im Netz verfügt, ergibt sich ein Overhead von $\frac{1}{32} \approx 3\%$ innerhalb des Flit-Kopfes. Zusätzlich sei an dieser Stelle erwähnt, dass in der Regel in einem Flit-Kopf Bits zur *freien* Verwendung reserviert werden. Zum anderen werden Flit-Köpfe mit Steuerungsinformationen versehen und selten mit den eigentlichen Nutzdaten. Es ist also möglich für Heartbeat-Nachrichten eines dieser freien Bits zu verwenden, womit im Prinzip keine zusätzlichen Kosten entstehen.

4.5.2. Polling

Im Gegensatz zum Push-Mechanismus, bei dem die Prozessorkerne selbständig Heartbeat-Nachrichten an die FDU senden, warten die Kerne beim Polling auf eine entsprechende Anfrage der FDU. Da die Anfragen der FDU bereits Teil des Lokalisierungsverfahrens sind, werden auch diese mit den Heartbeat-üblichen Prioritäten durch das Netz übermittelt und müssen deshalb auch in die Kostenbewertung mit aufgenommen werden.

$$K_{Polling} = 2 \times \sum_{\forall c \epsilon C} MD(c) \times LB$$

Die Gleichung beinhaltet die jeweilige Distanz von Kern c zur FDU und die Bandbreite einer Verbindungsleitung pro Takt LB. Da sowohl der Hinweg, als auch der Rückweg berücksichtigt werden muss, wird das Ergebnis aller Kerne mit dem Faktor 2 multipliziert.

4.5.3. Überlappungen

Das Volumen der Überlappung hängt davon ab, ob die Cluster in einander verschoben werden, ohne dabei die Größe des individuellen Clusters zu verändern (Verschmelzung), oder ob durch die Überlappung ein Cluster größer wird (Expansion). Für den ersten Fall ergibt sich kein weiteres Datenvolumen, da sich weder die Zahl der Heartbeat-Nachrichten ändert, noch die Wegstrecken kürzer oder länger werden. Im zweiten Fall der Expansion hingegen werden dem Cluster neue Kerne hinzugefügt, was einerseits die Zahl der Heartbeat-Nachrichten innerhalb des Clusters erhöht und andererseits den Weg

der neuen Kerne zur FDU verlängert. Die Berechnung des zusätzlichen Overheads kann analog zum Polling durchgeführt werden. Für die neu hinzugefügten Kerne c' werden die Distanzen zur FDU berechnet und mit der Verbindungsbreite multipliziert:

$$K_{Overlap} = 2 \times \sum_{\forall c' \in C} MD(c') \times LB$$

4.5.4. Künstliche Verzögerungen

Der zusätzliche Overhead, welcher durch die künstlichen Umwege entsteht, ist abhängig von dem Ort an dem der Fehler auftrat. Am Rand eines Clusters werden die Verbindungsleitungen weniger häufig von Heartbeat-Nachrichten verwendet, als es vergleichsweise im Cluster-Zentrum der Fall ist. Tritt ein Fehler am Rand eines Cluster auf, sind also weniger Heartbeat-Nachrichten von dem Fehler und dem daraus resultierendem Umrouten betroffen.

Betrachtet man lediglich die Kosten einer einzelnen Heartbeat-Nachricht, sind diese konstant für ein einzelnes Umrouten. Bedingt durch die Gitterstruktur des Netzes erfolgt das Umrouten durch zwei zusätzliche Hops pro Fehler:

$$K_{Bypass} = 2 \times LB \times HB$$

4.6. Multiple Fehler

Wie in der Einleitung dieses Kapitels bereits erwähnt wurde, basieren die bisherigen Ergebnisse auf einzeln auftretenden Fehlern der Hardware innerhalb des Kommunikationsnetzes. Studien belegen jedoch, dass multiple Fehler in Hardware nicht selten sind [Schroeder 2009; Schroeder und Gibson 2010; Nightingale 2011]. Dennoch sorgen Fehlertoleranztechniken dafür, dass die fehlerhafte Hardware nicht immer sofort ausgetauscht werden muss. Gerade bei transienten Fehlern erlauben Fehlerkorrekturtechniken einen Weiterbetrieb der Systeme bei moderatem Leistungsverlust. Dies ist nach Ansicht von [Schroeder und Gibson 2010] gerade bei Servern wichtig, da der Austausch der fehlerhaften Komponente und dem dadurch erzeugten Ausfall eines Servers unwirtschaftlich sein kann. Demnach sollte abgewogen werden, ob eine höhere Rate transienter Fehler tolerierbar ist oder ob das defekte Modul direkt ausgetauscht wird.

Unglücklicherweise konnten die Arbeiten keine permanenten Fehler bewerten, da diese in der Regel sofort zum Austausch der fehlerhaften Hardware geführt haben. Es ist also nicht trivial zu zeigen, wie häufig sich multiple permanente Fehler auf der Hardware manifestiert haben. Dennoch zeigen die Projektionen der *International Technology Roadmap of Semiconductors (ITRS)* [ITRS 2007; ITRS 2011] einen Anstieg der Fehlerwahrscheinlichkeit innerhalb zukünftiger Computer-Systeme. Als Einflussfaktor der

steigenden Fehlerraten nennt die ITRS sowohl die stetige Verkleinerung der Strukturbreiten, welche die Fehlerwahrscheinlichkeit pro Transistor erhöht, als auch die steigende Anzahl der Transistoren auf dem Chip. Ähnliches muss auch von den Verbindungsleitungen des prozessorinternen Kommunikationsnetzes erwartet werden, da diese gleichermaßen von den Problemen der kleiner werdenden Strukturgrößen betroffen sind. Es muss also davon ausgegangen werden, dass bei langer Systemlaufzeit vermehrt Fehler auch im Kommunikationsnetz auftreten können.

Da das Verfahren zur Fehlerlokalisierung den Ort eines Fehlers aus impliziten Daten ableitet, deren Informationen möglicherweise widersprüchlich sein können, ist es notwendig, diese Widersprüche zu identifizieren. Die bisherigen Ergebnisse zeigen zwar, dass ein einzeln auftretender Fehler im Netz zuverlässig lokalisiert werden kann, doch gilt dies nicht mehr bei multiplen Fehlern im Netz. Die nachfolgende Diskussion zeigt, dass Fehler in der Tat andere Fehler maskieren können. Zudem existiert ein Effekt, der dafür sorgt, dass bei der Auswertung der gesammelten Informationen ein Trugbild entsteht, sodass ein Fehler diagnostiziert wird, der in Wirklichkeit nicht existiert. Grund für diese Schwächen des Lokalisierungsverfahrens sind spezielle Kombinationen aus räumlichen und zeitlichen Erscheinungsmustern der multiplen Fehler.

4.6.1. Analysemethode

Für die Analyse wurden zeitliche und räumliche Fehlermuster miteinander kombiniert auf das Netz angewendet. Dazu wurden die zwei zeitlichen Muster *simultan* und *sukzessiv* definiert. Die Unterscheidung beider Muster ist notwendig, da die FDU erst alle Heartbeat-Nachrichten einer TDMA-Runde sammeln muss, um anschließend die Suche nach den lokalen Maxima durchführen zu können. Treten die Fehler innerhalb einer TDMA-Runde auf, sind diese als simultan definiert. Analog dazu ist das Auftreten der Fehler in unterschiedlichen TDMA-Runden als sukzessiv definiert. In Kombination mit räumlichen Mustern, bewirken die zeitlichen Muster unterschiedliche Ergebnisse bei der Lokalisierung. Deshalb befasst sich die nachfolgende Diskussion jeweils getrennt mit beiden zeitlichen Mustern.

Die räumlichen Muster bestehen aus zwei Fehlern und wurden systematisch auf das Netz permutiert. Ausgewertet wurden die Ergebnisse der Lokalisierung anhand der aus dem vorherigen Kapitel bekannten FDU internen Statusmatrix M^F. Zeigte die Matrix den erwarteten Zustand mit allen lokalisierten Fehlern, wurde das Muster als unproblematisch klassifiziert. Analog wurden Abweichungen vom erwarteten Zustand als problematisch deklariert und analysiert. Für die Analyse wurde zusätzlich die Matrix der verdächtigten Netzkomponenten M^V herangezogen, um den Grund für die Abweichung in der Lokalisierung zu ermitteln. Während der Analyse konnten wiederkehrende räumliche Muster erkannt werden, die jeweils das gleiche Problem bei der Lokalisierung verursachten. Wo es möglich war, wurden diese Muster zu Gruppen zusammengefasst und diskutiert.

Aus den problematischen Mustern mit zwei Fehlern lassen sich leicht komplexere Muster erzeugen, indem man diese miteinander kombiniert. Die komplexeren Muster sind dann ebenfalls problematisch bei der Lokalisierung. Aus dem Grund kann auf die Permutationen mit höherer Fehlerzahl verzichtet werden, da sich die Ergebnisse in der Lokalisierung dadurch nicht ändern.

4.6.2. Anwendung der Fehlermuster auf das Netz

Wie eingangs beschrieben, unterscheidet die Diskussion die zeitlichen Muster zwischen simultan und sukzessiv. Die Untersuchung beginnt daher zunächst mit der Anwendung von simultanen Fehlern und diskutiert räumliche Muster, welche als problematisch eingestuft wurden. Im Anschluss daran wird die Untersuchung in Bezug auf sukzessive Fehler wiederholt und die Ergebnisse diskutiert.

4.6.2.1. Simultan auftretende Fehler im Netz

Die Suche nach problematischen Mustern mit simultan auftretenden Fehlern lieferte drei Gruppen, in die sich die räumlichen Muster einordnen ließen:

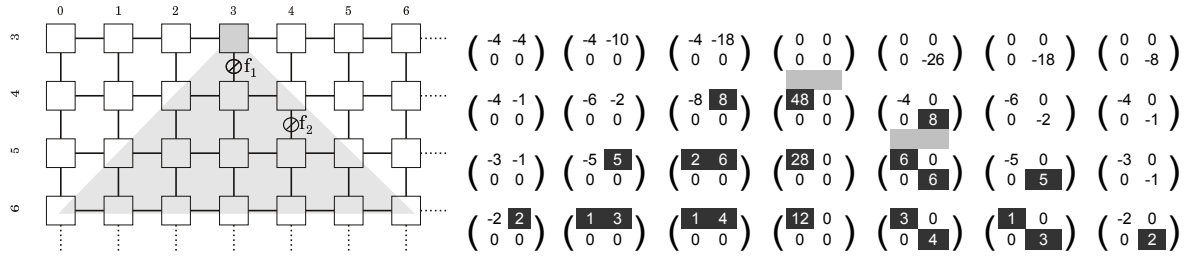
(i) Fehler in der direkten Umgebung der FDU.

(ii) Fehler mit gemeinsamen Teilpfaden.

(iii) Eng zusammenliegende Fehler.

Bei Fehlermustern der Gruppe *»Fehler in direkter Umgebung der FDU«* kommt es zu einer Fehlermaskierung, bei dem ein Fehler an einer bestimmten Position im Netz die Lokalisierung weiterer Fehler verhindert. Das Problem hierbei ist die räumliche Nähe einer fehlerhaften Verbindungsleitung zur FDU. Denn je näher eine fehlerhafte Komponente der FDU ist, desto mehr Heartbeat-Nachrichten werden von dieser Komponente verzögert. Dadurch entsteht ein toter Winkel (graues Dreieck in Abbildung 4.6(a)), der jeden weiteren Fehler innerhalb dieses toten Winkels verbirgt.

Um diese Problematik zu verdeutlichen, wurden für das Netz in Abbildung 4.6(a) die Verbindungsleitungen $R_{(3,4)} \overset{N}{\rightarrow} R_{(3,3)}$ (f_1) und $R_{(4,5)} \overset{N}{\rightarrow} R_{(4,4)}$ (f_2) als fehlerhaft angenommen. In der Abbildung 4.6(b) ist der zugehörige Ausschnitt der Matrix M^V dargestellt. Die schwarz hinterlegten Werte deuten die während der Überwachungsrunde verdächtigten Komponenten an. Sucht man in dieser Matrix nach den lokalen Maxima, gelingt dies jedoch nur für die Verbindungsleitung $R_{(3,4)} \overset{N}{\rightarrow} R_{(3,3)}$. Wie zu erkennen ist, erstreckt sich damit ein toter Winkel von der FDU ausgehend diagonal bis zum unteren Rand des Clusters. Alle innerhalb dieses Bereichs abgesetzten Heartbeat-Nachrichten sind also von Verzögerungen durch f_1 oder f_2 (graue Rechtecke in der Matrix) betroffen. Verschiebe man f_2 auf eine beliebige andere Komponente innerhalb des toten Winkels, ändert sich nichts an der resultierenden Matrix.



(a) Schematische Darstellung des Fehlermusters $f_1 \leftrightarrow f_2$.

(b) Zustand von M^V für die Fehler $f_1 \leftrightarrow f_2$, nach Erhalt aller Heartbeat-Nachrichten.

Abb. 4.6.: Problematisches räumliches Fehlermuster, bei denen ein Fehler direkt an der FDU positioniert ist und simultan mit anderen Fehlern auftrat. Die Suche nach den lokalen Maxima scheitert wegen des toten Winkels ausgehend von f_1.

Ferner kommt hinzu, dass alle Heartbeat-Nachrichten aus dem toten Winkel bei der Aktualisierung des TDMA-Schemas und der erwarteten Ankunftszeiten entsprechend auf den Fehler f_1 angepasst werden. Damit ist es auch nicht möglich durch nachfolgende Überwachungsrunden weitere Fehler im toten Winkel aufzuspüren. Dies macht diese Fehlergruppe kritisch.

Eine kostengünstige Lösung dieses Problems liegt darin, die FDU von ihrem aktuellen Prozessorkern auf einen anderen Kern migrieren zu lassen. Da die FDU aus Software besteht, ist dies leicht möglich. Zu beachten ist jedoch, dass zunächst die überwachten Prozessorkerne das Absetzen der Heartbeat-Nachrichten stoppen und auf neue Konfigurationsnachrichten seitens der FDU warten. Nachdem die letzten Heartbeat-Nachrichten der aktuellen Überwachungsrunde bei der FDU eingetroffen sind, wird der Zustand der FDU eingefroren und auf einen anderen Kern übertragen. Dabei kann der Inhalt der Statusmatrizen verworfen werden, da sich durch das Migrieren auch die Distanzen der Prozessorkerne zur FDU und damit die Ankunftszeiten ändern.

Wie schnell sich mit dieser Lösung die toten Winkel dieses Fehlermusters auflösen lassen, zeigt die Abbildung 4.7. Es handelt sich dabei um die resultierende Matrix M^V nach einer vollständigen Überwachungsrunde. Das angewendete Fehlermuster ist dasselbe wie zuvor beschrieben. Diesmal jedoch wurde die FDU vor der Lokalisierung um einen Platz im Netz nach oben und rechts verschoben. Der Fehler f_1 befindet sich somit nicht mehr

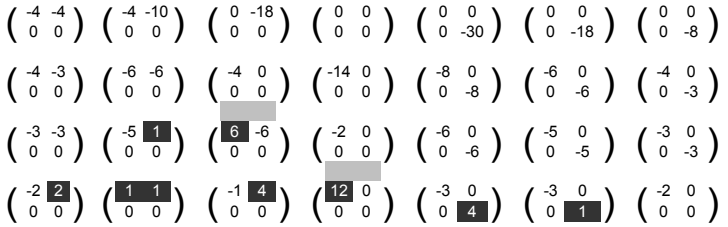
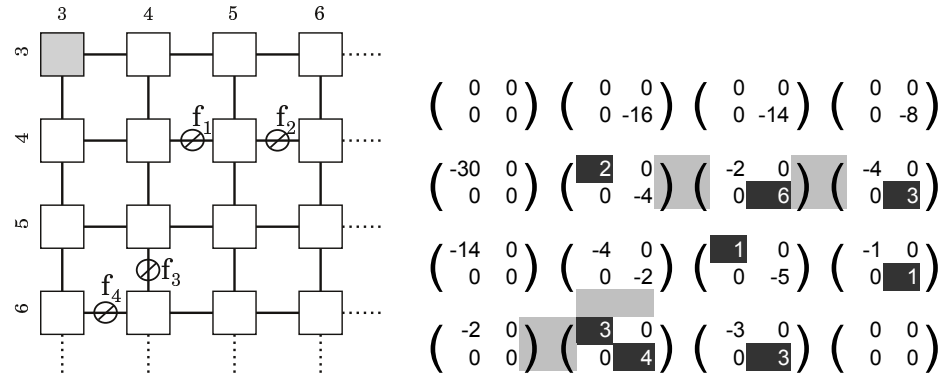


Abb. 4.7.: Zustand der Matrix M^V nach der Migration der FDU



(a) Schematische Darstellung von "auf einer Linie" ($f_1 \leftrightarrow f_2$) und "orthogonal zu einander" ($f_3 \leftrightarrow f_4$) liegender Fehler.

(b) Zugehöriger Zustand der Matrix M^S für die Fehlermuster $f_1 \leftrightarrow f_2$ und $f_3 \leftrightarrow f_4$ nach Erhalt aller Heartbeat-Nachrichten.

Abb. 4.8.: Problematische räumliche Fehlermuster, bei denen die Fehler eng zusammenliegen.

in direkter Nähe zur FDU und die Wirkung des toten Winkels hat bereits jetzt soweit nachgelassen, dass beide Fehler problemlos durch die Suche nach den lokalen Maxima lokalisierbar sind.

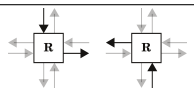
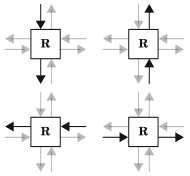
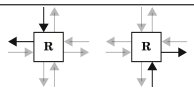
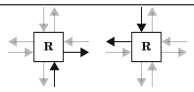
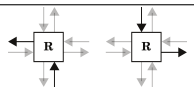
»Eng zusammenliegende Fehler« können dazu führen, dass mit der Suche nach den lokalen Maxima nicht alle Fehler gefunden werden. Auch hier erzeugt eine fehlerhafte Komponente, welche sich näher an der FDU befindet einen, wenn auch sehr kleinen, toten Winkel. Der Effekt kann bei zwei unterschiedlichen Ausprägungen beobachtet werden:

(i) Die fehlerhaften Komponenten liegen auf einer geraden Linie zueinander.

(ii) Die fehlerhaften Komponenten liegen in einem bestimmten Cluster-Quadranten orthogonal zueinander.

Abbildung 4.8(a) zeigt beide Ausprägungen dieser Fehlermuster anhand von zwei Beispielmustern; $f_1 \leftrightarrow f_2$ und $f_3 \leftrightarrow f_4$. In Abbildung 4.8(b) ist der resultierende Zustand der Matrix M^V dargestellt. Wie auch schon beim Fehlernmuster *»in der Nähe der FDU«*, kann aus der Matrix M^V entnommen werden, dass nicht alle lokalen Maxima gefunden werden können, die auf eine fehlerhafte Komponente hindeuten.

Während der tote Winkel bei allen Mustern auftritt, deren fehlerhaften Komponenten sich *»in einer Linie«* befinden, sind die orthogonal ausgerichteten Fehlermuster nur dann problematisch, wenn das Muster eine bestimmte Ausrichtung zur FDU aufweist und sich zusätzlich in einem bestimmten Quadranten befindet. Welche Ausrichtung im jeweiligen Quadranten problematisch ist, wurde in Tabelle 4.1 zusammengefasst. Die Tabelle beinhaltet die vier Quadranten des FDU-Clusters (graue Rechtecke) mit jeweils der FDU (schwarzes Quadrat) im Zentrum. Bei den orthogonalen Mustern werden zusätzlich die Senderichtungen der Heartbeat-Nachrichten unterschieden. Hier zeigt sich, dass nur die

Quadrant	Orthogonal	In einer Linie
		
		
		
		

Tab. 4.1.: Problematische Fehlermuster bei direkt benachbarten Netzkomponenten. Je nach Quadrant gibt es Muster (schwarze Pfeile), die eine Lokalisierung aller Fehler verhindern.

Verbindungsleitungen der Router von dem Muster betroffen sind, welche den Weg einer Heartbeat-Nachricht zum Ziel verkürzen. Die Position innerhalb eines Quadranten hingegen ist dabei unerheblich. Der Effekt tritt immer ein, wenn eines dieser orthogonalen Muster entsprechend auf das Netz angewendet wird. Entsprechen die Fehler dem Muster *»in einer Linie«* gibt es keine Unterscheidung der Quadranten. Dieser Effekt ist unabhängig bezüglich Position und Ausrichtung im Cluster und verhindert in jedem Fall die Lokalisierung des Fehlers mit der höheren Distanz zur FDU.

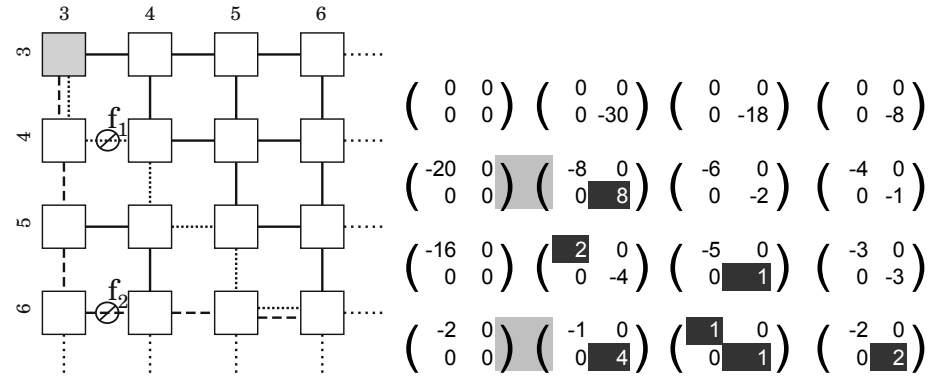
Grundsätzlich ist dieses Problem der Lokalisierung mit dem des Musters *»in direkter Umgebung der FDU«* vergleichbar. Auch hier sorgt ein Fehler für einen toten Winkel und verhindert die erfolgreiche Suche nach beliebigen Fehlern im Netz. Da auch hier Fehler maskiert werden, ist diese Gruppe als kritisch einzuordnen.

»Fehler auf gemeinsamen Teilpfaden« verursachen Trugbilder und sorgen dafür, dass Netzkomponenten fälschlich als defekt klassifiziert werden. Es handelt sich also um das genaue Gegenteil zur Fehlermaskierung durch die vorhergehenden Muster. Wie in Abschnitt 4.2 bereits beschrieben wurde, sollen fälschlich verdächtigte Netzkomponenten durch die Verwendung verschiedener Routing-Strategien für Heartbeat-Nachrichten wieder rehabilitiert werden. Dies scheitert jedoch bei multiplen Fehlern, wenn die folgenden Bedingungen erfüllt sind:

(i) Es gibt zwei Heartbeat-Nachrichten des gleichen Senders, die trotz unterschiedlicher Routing-Strategie, gemeinsame Teilpfade benutzen.

(ii) Im Netz existieren zwei Fehler, die jeweils auf einem der beiden tatsächlich disjunkten Pfade liegen.

(iii) Ein gemeinsamer Teilpfad befindet sich zwischen dem Sender und dem Fehler.



(a) Schematische Darstellung des Fehlermuster $f_1 \leftrightarrow f_2$

(b) Zustand für $f_1 \leftrightarrow f_2$

Abb. 4.9.: Muster: Fehler auf gemeinsamen Teilpfaden.

Sind diese Eigenschaften erfüllt, kann der Teilpfad mit der größeren Distanz zur FDU nicht rehabilitiert werden und erzeugt ein *Phantom-Fehler*. Abbildung 4.9(a) veranschaulicht dies in einem Beispiel-Szenario. Im gegebenen Netz werden zwei Verbindungsleitungen an den Stellen f_1 und f_2 als fehlerhaft angenommen. Der Prozessorkern am Router $R_{(6,6)}$ verschickt zwei Heartbeat-Nachrichten mit jeweils unterschiedlichen Routing-Strategien[4]. In der Abbildung sind die resultierenden Pfade der Nachrichten als gestrichelte Pfade dargestellt. In diesem Beispiel sorgen die Routing-Strategien XY- (grob gestrichelt) und Staircase-Routing (fein gestrichelt) dafür, dass auf den Verbindungsleitungen $R_{(6,6)} \overset{W}{\rightarrow} R_{(5,6)}$ und $R_{(3,4)} \overset{N}{\rightarrow} R_{(3,3)}$ gemeinsame Teilpfade der entsprechenden Heartbeat-Nachrichten entstehen. In Abbildung 4.9(b) ist die Matrix M^V nach einer vollständigen Überwachungsrunde dargestellt. Nach der Suche der lokalen Maxima wird, neben den Fehlern f_1 und f_2, ein weiterer Fehler der Verbindungsleitung $R_{(6,6)} \overset{W}{\rightarrow} R_{(5,6)}$ ermittelt. Letzterer Fehler ist demnach ein Phantom-Fehler der nicht wirklich existiert.

Die Fehler-Gruppe *»auf gemeinsamen Teilpfaden«* hat einen schwachen Einfluss auf die Lokalisierung. Das im Beispiel gezeigte Ergebnis zeigt zwar eine Fehlklassifizierung einer Netzkomponente, jedoch hat dies keinen direkten Einfluss auf die tatsächliche Leistungsfähigkeit des Netzes. Die Verbindungsleitung $R_{(6,6)} \overset{W}{\rightarrow} R_{(5,6)}$ wird auch weiterhin vom Router genutzt, um Nachrichten aller Arten zu übertragen. Allein die FDU-interne Repräsentation des Netzes ist in diesem Fall nicht zu 100% akkurat und beinhaltet eine pessimistische Annahme über den Zustand des Kommunikationsnetzes. Zudem besteht die Möglichkeit, dass eine Migration der FDU auf einen anderen Prozessorkern die Pfade der betroffenen Heartbeat-Nachrichten ändert und damit die Fehldiagnose korrigiert.

[4]Die Beachtung der übrigen Heartbeat-Nachrichten anderer Kerne kann an dieser Stelle vernachlässigt werden, da an der Position des Phantom-Fehlers nur die Heartbeat-Nachrichten des Kerns $C_{(6,6)}$ verarbeitet werden.

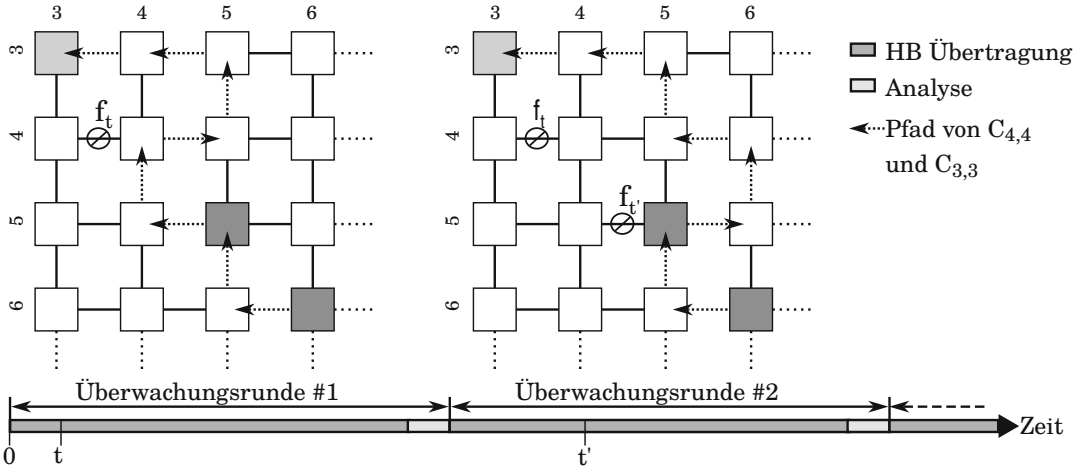


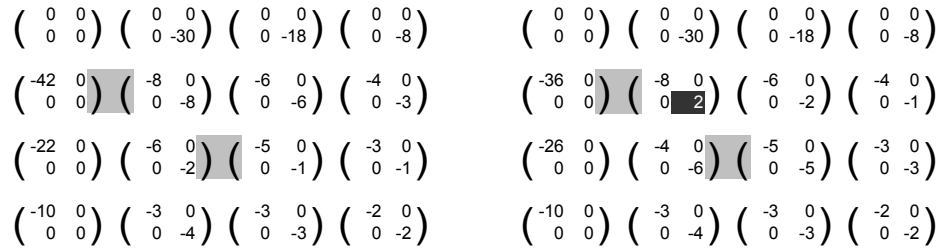
Abb. 4.10.: Beispiel eines sukzessiven Fehlermusters, bei dem die Lokalisierung von $f_{t'}$ auf Grund der Maskierung von f_t nicht möglich ist.

4.6.2.2. Sukzessive auftretene Fehler

Zu den oben aufgeführten räumlichen Mustern gibt es bei sukzessiv auftretenden Fehlern eine Besonderheit. Treten die Fehler nacheinander im Netz auf, bleiben zum Teil die Effekte eines toten Winkels oder der Phantom-Fehler aus. Berücksichtig man ferner die Reihenfolge in denen die Fehler auftreten, können sogar alle räumliche Muster unproblematisch sein. Die Auswirkung sukzessiver Fehler wird nachfolgend anhand eines Beispiels verdeutlicht. Die Diskussion, wie sich dieses zeitliche Muster auf Effekte der räumlichen Muster auswirkt, findet im Zusammenhang mit der Quantifizierung im nächsten Abschnitt statt.

Zur Veranschaulichung wird das in Abbildung 4.10 illustrierte Szenario zweier sukzessive auftretender Fehler in einem FDU-Cluster angenommen. Aus zeitlicher Sicht ist der Ablauf des Szenarios in zwei Phasen unterteilt. Jede Phase besteht jeweils aus einer vollständigen Überwachungsrunde, in denen sowohl die Übertragung aller Heartbeat-Nachrichten realisiert ist, als auch die Auswertung der Ankunftszeiten zusammen mit der Anpassung des TDMA-Schemas. Die Pfade der Heartbeat-Nachrichten resultieren in diesem Beispiel aus der Staircase-Strategie. Für blockierte Heartbeat-Nachrichten wurde zusätzlich das Protokoll der künstlichen Verzögerung angewendet und für beide Phasen durch gestrichelte Pfeile angedeutet.

In Phase #1 wurde der Fehler f_t auf das Netz angewendet. Durch die gestrichelten Pfeile wird ersichtlich, wie das Protokoll der künstlichen Verzögerung die Ankunft der Heartbeat-Nachrichten der Prozessorkerne $C_{(4,4)}$, $C_{(5,5)}$ und $C_{(6,6)}$ durch einen Umweg verzögert. In der Analyse der ersten Phase werden nun die abweichenden Ankunftszeiten dieser Nachrichten ausgewertet und der Fehler f_t lokalisiert. Da die Pfade der betroffenen Heartbeat-Nachrichten auf Dauer dieser Route zur FDU folgen werden, wird das TDMA-Schema entsprechend angepasst, um die gegenseitige Isolation der Heartbeat-



(a) $f_t \rightarrow f_{t'}$
Lokalisierung von $f_{t'}$ fehlgeschlagen

(b) $f_{t'} \rightarrow f_t$
Lokalisierung von f_t erfolgreich.

Abb. 4.11.: Resultierende Matrizen der verdächtigten Netzkomponten für die zeitlichen Fehlermuster (a) $f_t \rightarrow f_{t'}$ und (b) $f_{t'} \rightarrow f_t$ am Ende der zweiten Überwachungsrunde.

Nachrichten zu gewährleisten. Damit ist die Lokalisierung zu Beginn der Phase #2 wieder korrekt konfiguriert.

In der folgenden Phase #2 wird ein weiterer Fehler $f_{t'}$ auf das Netz angewendet. Auch hier greift das Protokoll der künstlichen Verzögerung und leitet die Heartbeat-Nachrichten diesmal am Router $R_{(5,5)}$ um. Zwar entstehen auch in diesem Fall wieder Verzögerungen der Heartbeat-Nachrichten, doch erwartet die FDU diese Verzögerung durch die vormals durchgeführte Anpassung des TDMA-Schemas. Die daraus resultierende Matrix M^V ist in Abbildung 4.11(a)[5] dargestellt. Da alle Werte der Matrix ≤ 0 sind, gibt es für die FDU keinen Hinweis darauf, dass die Nachrichten diesmal vom Fehler $f_{t'}$ verzögert wurden.

Entscheidend für das Auftreten eines toten Winkels ist jedoch die Reihenfolge, in der die Fehler auftreten. Kehrt man die Reihenfolge aus dem obigen Beispiel bezüglich der Fehler um, entsteht am Ende der zweiten Überwachungsrunde die Matrix aus Abbildung 4.11(b). Der schwarz hinterlegte positive Wert ergibt bei der Suche nach einem lokalen Maximum die Position des Fehlers f_t. Grund für die erfolgreiche Lokalisierung ist in diesem Fall, dass mit Hilfe der Heartbeat-Nachrichten von $C_{(5,5)}$ und $C_{(6,6)}$, der Fehler $f_{t'}$ zunächst in Phase #1 lokalisiert werden konnte. Das nachfolgende Auftreten von f_t in Phase #2 wirkt nun nur noch auf die Verzögerung der Heartbeat-Nachrichten von $C_{(4,4)}$. Da die Heartbeat-Nachrichten von $C_{(4,4)}$ zuvor ohne Verzögerung eintrafen, kann damit ebenfalls der Fehler f_t lokalisiert werden.

Obwohl die Heartbeat-Nachrichten in dem oben stehenden Szenario der Staircase-Strategie folgen, ist der Effekt der Maskierung nicht auf diesen Algorithmus beschränkt. Tatsächlich lassen sich für alle vier der hier verwendeten Routing-Algorithmen ähnliche Muster mit dem gleichen Resultat erzeugen.

[5] Da die injizierten Fehler innerhalb der Phase #1 in beiden Fällen lokalisiert werden konnten, wurde der Zustand der Matrix M^V dieser Phase #1 in der Abbildung 4.11 weggelassen.

4.6.3. Quantifizierung

Zum Abschluss der Untersuchung multipler Fehler wird geklärt, wie groß der Anteil der Fehlermuster ist, die als problematisch bei der Lokalisierung gelten. Damit wird die Leistungsfähigkeit der hier vorgestellten Lokalisierung für multiple Fehler quantitativ aufgezeigt. Die Grundlage der Quantifizierung besteht aus der Auszählung der jeweiligen problematischen Fehlermuster. Die hier präsentierten Ergebnisse entsprechen dabei dem prozentualen Anteil der oben diskutierten Fehlergruppen. Zusätzlich wurden drei verschiedene Cluster-Größen ($5 \times 5, 7 \times 5$ und 7×7) bei der Quantifizierung zugrunde gelegt und in Tabelle 4.2 zusammengefasst.

Positiv zu bewerten ist, dass das als sehr kritisch identifizierte Fehlermuster *»in direkter Umgebung der FDU«* im schlechtesten Fall in etwa $1,5\%$ aller Fehlermuster auftreten kann. Unter Berücksichtigung der Reihenfolge in der die Fehler auftreten sinkt dieser Wert weiter auf $0,725\%$.

Deutlich größer hingegen ist der Anteil des Fehlermusters *»eng zusammenliegende Fehler«* im Netz. Die kompakte Form dieses räumlichen Musters, sowie die beiden verschiedenen Ausprägungen *»orthogonal«* und *»in einer Linie«* ergeben zusammen einen Anteil von $6,522\%$ bei simultanen und bis zu $1,631\%$ bei sukzessiven Fehlermustern.

Das räumliche Muster *»Fehler mit gemeinsamen Teilpfaden«* fällt durch zwei Besonderheiten auf. Zum Einen gibt es bei kleinen Cluster-Größen keine Möglichkeit dieses Muster zu platzieren, da es beim Platzieren einen Mindestabstand der beiden Fehler voraussetzt. Wird der Mindestabstand nicht eingehalten, entsteht beispielsweise das Muster *»eng zusammenliegende Fehler«*. Zum Anderen ist das Muster nur dann problematisch, wenn die Fehler simultan im Netz auftreten. Bei sukzessiven Fehlern entsteht der Phantom-Fehler hingegen nicht.

	simultan			**sukzessiv**		
	5×5	7×5	7×7	5×5	7×5	7×7
I.d.U.[1)]	1,449	0,713	0,355	0,725	0,357	0,177
E.z.F.[2)]	6,522	6,061	5,674	1,631	1,515	1,419
A.g.T.[3)]	–	–	3,901	–	–	–
Gesamt	7,971	6,774	9,929	2,356	1,872	1,596

Alle Werte in %.
–: Keine problematischen Fehlermuster dieses Typs für die gegebene Clustergröße gefunden.
[1)] Muster: **I**n **d**irekter **U**mgebung der FDU
[2)] Muster: **E**ng **z**usammenliegende **F**ehler
[3)] Muster: **A**uf **g**emeinsamen **T**eilpfaden

Tab. 4.2.: Zusammenfassung aller Fehlermuster für die Quadrantengröße 5×5, 7×5 und 7×7

Alle problematischen Fehlermuster zusammengefasst zeigen, bei sukzessive auftretenden Fehlern, dass etwa $2,4\%$ der Fehlermuster zu einer Fehlermaskierung oder einem Phantom-Fehler führen. Beim simultanen Auftreten vergrößert sich der Anteil der pro-

blematischen Fehler auf etwa 10%. Dennoch kann mit wachsender Cluster-Größe eine stetige Verringerung des prozentualen Anteils der kritischen Fehlermuster beobachtet werden.

Zusätzlich sei an dieser Stelle darauf hingewiesen, dass simultane Fehler im Netz zwar grundsätzlich möglich sind, doch kann erwartet werden, dass die Wahrscheinlichkeit für multiple Fehler sukzessiver Natur im Netz deutlich höher ist. Diese Annahme wird durch die Berücksichtigung der Fehlerursachen gestützt. Die Gründe für einen permanenten Fehler sind beispielsweise der Verschleiß durch Elektromigration und einseitige oder thermale Belastungen[6] [Wang und Filippi 2001; Borkar 2005; Furber 2008]. Abgesehen von durch physische Einwirkungen verursachte Fehler, kann angenommen werden, dass mit der Verkleinerung der Strukturgrößen vermehrt permanente Fehler auftreten werden. Diese werden sich jedoch auf dem Chip verteilen und zu unterschiedlichen Zeitpunkten erscheinen.

4.7. Erweiterung auf den Torus

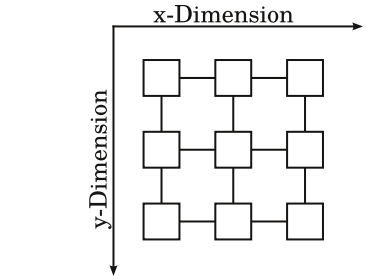
Toroidale Netztopologien werden aus Gründen der Leistungsfähigkeit und der Fehlertoleranz häufig im Bereich der Computernetze eingesetzt. Durch ihren mehrdimensionalen Aufbau und eine entsprechende Routing-Strategie ermöglichen sie nicht nur eine bessere Lastverteilung des Datenverkehrs im Kommunikationsnetz, sondern erhöhen gleichzeitig die Anzahl redundanter Pfade durch das Netz. Großrechenanlagen setzen beispielsweise auf einen dreidimensionalen Torus und adaptives Routing um Kommunikationsengpässe zu vermeiden und fehlerhafte Verbindungsleitungen zu kompensieren.

Für die Kommunikation auf einem Prozessor mit hunderten oder tausenden Kernen existieren jedoch erst wenige Lösungen, die auf einer toroidalen Topologie basieren. Tatsächlich existiert - neben wenigen Forschungsprozessoren - mit dem Kalray MPPA-256 zurzeit [Dinechin 2013a][7] nur ein kommerziell verfügbarer Prozessor, dessen internes Kommunikationsnetz auf einem Torus basiert. Grund dafür dürften technische Herausforderungen einer on-Chip Implementierung sein, wie die unterschiedlichen Signallaufzeiten zwischen den Routern, den längeren Verbindungsleitungen [Ludovici 2009] und der Gefahr durch Deadlocks [Dally und Towles 2003].

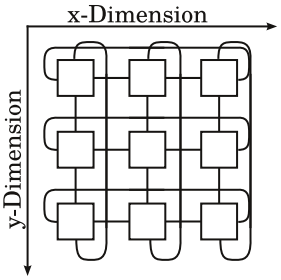
Verglichen mit einer 2D-Gitter Topologie, ist der Torus mit seinen zusätzlichen Verbindungsleitungen dennoch aus Sicht der Fehlertoleranz sehr attraktiv. Die zusätzliche Redundanz in der Kommunikationsinfrastruktur erhöht die Ausfallsicherheit bei den Kernen an den Rändern und in den Ecken des Prozessors. Wie bereits im vorherigen Kapitel beschrieben wurde, stellen gerade diese Kerne zusätzliche Anforderungen an das Verfahren

[6] Auf einem Prozessor existieren verschiedene Bereiche, welche je nach Verwendung während des Betriebs unterschiedlich heiß werden können. Die Pipeline eines Prozessorkerns beispielsweise wird in der Regel deutlich heißer, als es dessen Cache-Speicher[Borkar 2005].

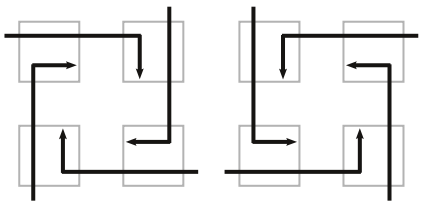
[7] Stand: 15.01.2014



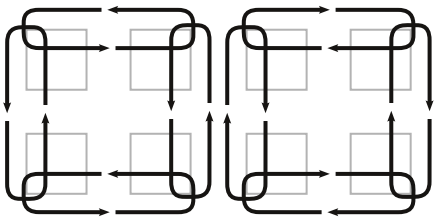
(a) 2D-Gitter



(b) Torus



(c) Zyklische Abhängigkeit eines 2D-Gitters mit vier Kernen, die zu einem Deadlock führt.



(d) Zyklische Abhängigkeit eines Torus mit vier Kernen, die zu einem Deadlock führt.

Abb. 4.12.: Verschiedene Netztopologien und zugehörige Abhängigkeitsgraphen

der Fehlerlokalisierung in einem 2D-Gitter. Daher werden in diesem Abschnitt Vor- und Nachteile einer toroidalen Netztopologie unter Berücksichtigung der Routing-Strategie und der Fehlerlokalisierung diskutiert.

4.7.1. Routing im Torus

Anders als ein 2D-Gitter besteht ein Torus aus mehreren Ringen, die mehrfach miteinander verbunden sein können. Zum Vergleich der beiden Topologien wurde in den Abbildungen 4.12(a) und (b) der Aufbau eines 2D-Gitters neben einer toroidalen Implementierung dargestellt. Der Aufbau beider Topologien unterscheidet sich im Wesentlichen nur durch die sogenannten *Wrap-Around Links* (WAL). Die Verbindungsleitungen verbinden jeweils sich gegenüberliegende Ränder des Prozessors miteinander und bilden damit Ringe. Die Router dieser Topologie sind also mit jeweils einem Ring in X-Richtung und Y-Richtung verbunden. Die Ringstruktur sorgt beim Routing von (Heartbeat-)Nachrichten jedoch für drei Probleme:

- Gefahr von Deadlocks.
- WALs werden nicht von den Routing-Strategien genutzt.
- Durch das Routing bedingte tote Winkel auf den WALs.

4.7.1.1. Gefahr durch Deadlocks

In einem Netz kommt es immer dann zu einem Deadlock, wenn Nachrichten in der Lage sind, zyklische Abhängigkeiten zu erzeugen. Für ein 2D-Gitter bedeutet das, dass sich die Abhängigkeiten immer über beide Dimensionen erstrecken müssen, um einen Kreis zu bilden (Abbildung 4.12(c)). Die zyklische Abhängigkeit kann in diesem Fall verhindert werden, indem nur eine Richtungsänderung pro Nachrichtenpfad zugelassen wird[8] und die erste Senderichtung vorgegeben ist. Beispielsweise werden Nachrichten mit der XY-Strategie zunächst immer erst in X-Richtung übertragen, bevor die Übertragungsrichtung einmalig auf die Y-Richtung geändert wird. Bei den Ringen des Torus hingegen, ergeben sich die zyklischen Abhängigkeiten bereits in einer einzelnen Dimension (Abbildung 4.12(d)). Durchbrochen werden diese Abhängigkeiten in toroidalen Netzen ebenfalls mit dem Einsatz von virtuellen Kanälen in Form von zusätzlichen Pufferspeichern im Router [Duato 2002; Dally und Towles 2003].

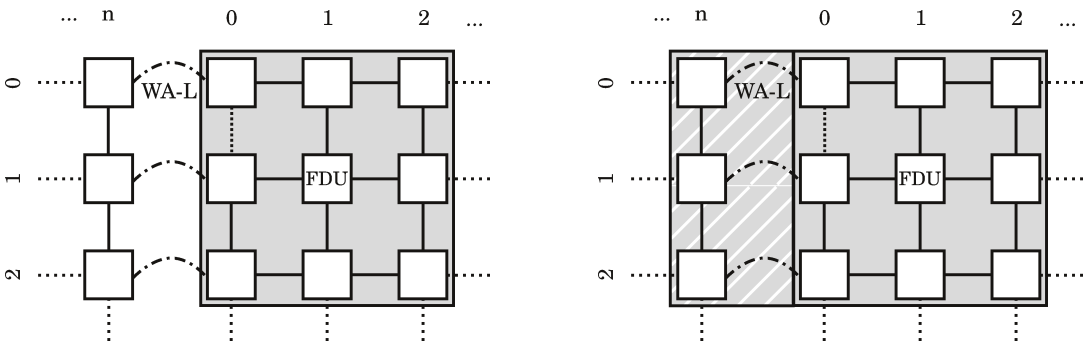
4.7.1.2. Ungenutzte Wrap-Around Links

Die Routing-Strategie muss in der Lage sein, die WALs gewinnbringend einzusetzen. Das bedeutet, die Routing-Logik muss entscheiden, in welche Richtung des Rings die Nachricht gesendet werden soll, um diese auf dem tatsächlich kürzesten Weg zu übermitteln. Die für die Fehlerlokalisierung notwendigen Routing-Strategien XY und Staircase (einschließlich ihrer invertierten Versionen aus Kapitel 3) können per se nicht die WALs des Torus nutzen. Das Routing-Schema dieser Verfahren ist weder auf mehr als zwei Dimensionen ausgelegt, noch kann es von einer ringähnlichen Topologie profitieren. Für den Torus wäre also eine Anpassung der Routing-Strategien im Router notwendig. Alternativ wird häufig auch das Quell-Routing eingesetzt, bei dem der sendende Kern die Route durch das Netz bereits vorplant und in speziellen Sektionen des Nachrichtenkopfes einträgt [Pande 2005b]. Damit muss ein Router entlang des Pfads keine eigene Entscheidung treffen und kann die Routing-Information direkt aus dem Nachrichtenkopf nutzen. Eine zwischenzeitliche Änderung des Netzes durch fehlerhafte Elemente kann jedoch damit nicht berücksichtigt werden und ist damit für die Lokalisierung ebenfalls ohne zusätzliche Anpassungen nicht verwendbar.

4.7.1.3. Durch das Routing bedingte tote Winkel

Fügt man die Anpassungen bezüglich der Routing-Strategien dem Router hinzu, in dem die Logik der Routing-Einheit erweitert wird, können zwar die Anwendungsnachrichten von den WALs profitieren, nicht aber die Heartbeat-Nachrichten. Denn die FDU liegt weiterhin im Zentrum des FDU-Clusters. Das bedeutet, dass sich alle Heartbeat-

[8] Deterministische Routing-Strategien nutzen weitere Turn-Modelle um auch Fehler im Netz zu tolerieren und gleichzeitig Deadlock-frei zu sein.



(a) Ohne Erweiterung des Cluster

(b) Mit Erweiterung des Cluster (Schraffierter Bereich)

Abb. 4.13.: Toter Winkel bei der Überwachung der Wrap-Around Links

Nachrichten vom Rand weg hin zur FDU bewegen. In Abbildung 4.13(a) ist hierzu ein Beispiel gegeben. Grau eingerahmt ist der gesamte FDU-Cluster, wobei sich die FDU auf dem Prozessorkern $C_{(1,1)}$ befindet. Durch die XY- und die Staircase-Strategie werden die Nachrichten direkt in Richtung FDU weitergeleitet. Die WALs werden nicht verwendet und somit auch nicht überwacht. Es entsteht also ein neuer toter Winkel, wie er bereits im Abschnitt 4.4.2 beschrieben wurde. Auch hier werden die Verbindungsleitungen direkt außerhalb des Clusters nicht verwendet. Glücklicherweise kann auch hier die gleiche Technik der überlappenden Cluster-Grenzen angewendet werden (siehe Abbildung 4.13(b)). Die Prozessorkerne der n-ten Spalte senden somit ihre Heartbeat-Nachrichten an zwei FDUs und lösen damit den toten Winkel auf.

4.7.2. Fehlerlokalisierung im Torus

Für die Fehlerlokalisierung ergibt sich mit dem Torus zunächst die Möglichkeit, die toten Winkel bei der Überwachung durch die WALs zu lösen, ohne dabei das Router-Design am Rand oder den Ecken des Prozessors weiter zu verändern (vergleiche auch Abschnitt 4.4.4). Im Falle einer künstlichen Verzögerung können die Heartbeat-Nachrichten über die WALs übertragen und von der anderen Seite aus auf dem kürzestem Weg zur FDU weitergeleitet werden. Abbildung 4.14 illustriert ein Beispiel dazu. Grau eingerahmt ist wieder der FDU-Cluster. Die Verbindungsleitung des Routers $R_{(0,0)} \overset{S}{\rightarrow} R_{(0,1)}$ wurde als fehlerhaft angenommen und die Heartbeat-Nachricht wird gemäß der künstlichen Verzögerung auf dem WAL zunächst zum Router $R_{(0,n)}$ und anschließend zum Router $R_{(1,n)}$ übertragen. Von hier aus ist die kürzeste Verbindung zur FDU der Weg über den zweiten WAL und wird demnach der Routing-Strategie favorisiert. Dies ist vor allem deshalb wichtig, da andernfalls die Heartbeat-Nachricht den Cluster verlassen würde und andere Cluster durchquert. Damit bestünde die Gefahr, dass sie mit anderen Heartbeat-Nachrichten der anderen Cluster kollidiert.

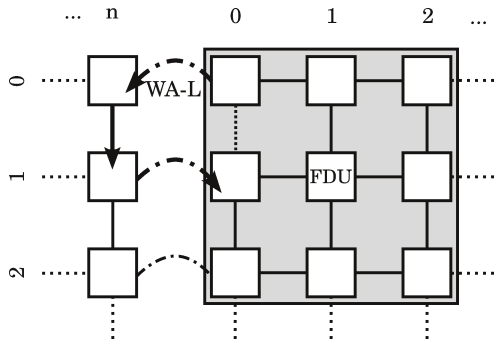


Abb. 4.14.: Ring-Abhängigkeiten durch Wrap-Around-Verbindungen in einem Torus

Die WALs haben jedoch durch ihre längere Wegstrecke von einem Prozessorrand zum anderen ein technisches Problem, welches in Verbindung mit der *Signaldämpfung* steht [Dally und Towles 2003; Ludovici 2009]. Überschreitet eine Verbindungsleitung eine kritische Länge nimmt die Stärke des Signals auf ihr quadratisch ab. Verstärkt wird dieser Effekt zusätzlich durch hohe Betriebsfrequenzen im Netz. Um dieses technische Problem für lange Verbindungsleitungen und hohe Taktraten zu lösen, wurden bereits Techniken wie die Signalverstärkung (*Link Performance Boosting*) oder das *Link Pipelining* vorgeschlagen.

Die Verstärker befinden sich auf einer langen Verbindungsleitung, zwischen zwei Routern. Sie nehmen das Signal vor dem kritischen Punkt der Signaldämpfung auf und leiten es *aufgefrischt* auf der Leitung weiter. Es ist möglich multiple Verstärker auf einer Verbindungsleitung zu platzieren, um auch sehr lange Verbindungsleitungen zu unterstützen.

Beim Link Pipelining werden auf Verbindungsleitungen zwischen zwei "weit" entfernten Routern Puffer-Speicher eingesetzt, die es mehreren Paketen ermöglicht "gleichzeitig" über eine Verbindungsleitung gesendet zu werden. Ist beispielsweise eine Verbindungsleitung so lang, dass innerhalb eines Taktes die Pakete nicht übertragen werden können, werden diese Puffer als *Zwischenhalt* bis zum nächsten Takt verwendet [Gebhardt 2011].

Das Verstärken der Signalstärke allein kann auf WALs wegen dessen Länge nicht immer ausreichend sein. Denn mit Voranschreiten der Verkleinerung der Transistoren und Verbindungsleitungen des Prozessors, nimmt auch die Signaldämpfung weiter zu. Der kritische Punkt an dem das Signal aufgefrischt werden muss, rückt immer näher an den Sender, was wiederum die Anzahl der Verstärker pro Link erhöht. Link Pipelining wird demnach eine mögliche Lösung sein, um einen Torus auch in zukünftigen Fertigungstechniken zu realisieren[Flich und Bertozzi 2010]. Dennoch müssen die unterschiedlichen Übertragungsraten der normalen Verbindungsleitungen und der WALs der FDU bekannt sein, damit das TDMA-Schema zum Absetzen der Heartbeat-Nachrichten korrekt berechnet werden kann.

4.7.3. Zusammenfassung

Das hier vorgestellte Verfahren zur Fehlerlokalisierung kann auch auf einer toroidalen Netztopologie angewendet werden. Dabei löst die toroidale Topologie nativ die Problematik der Überwachung der Eck-Router eines Prozessors. Praktisch ist außerdem, dass tote Winkel der WALs mit den gleichen Mitteln des Überlappens der Cluster behoben werden können.

Einhergehen jedoch neue Herausforderungen, die der Torus mit sich bringt. Neben den technischen Herausforderungen, stellt das Deadlock-Risiko eine Gefahr dar, dessen Lösung mit zusätzlicher Hardware im Router erkauft werden kann. Zusätzlich müssen die Routing-Strategien angepasst werden, damit Nachrichten gegebenenfalls Nutzen aus den WALs ziehen können. Die XY- und die Staircase-Strategie sind per se nicht dazu in der Lage. Ohne eine entsprechende Anpassung würde das Netz so genutzt werden, als sei es ein 2D-Gitter.

5. Zusammenfassung und Ausblick

Diese Dissertationsarbeit präsentierte eine fehlertolerante Architektur des TERAFLUX-Prozessors, die als Grundlage für eine nachrichtenbasierende Fehlererkennung dient. Zustandsinformationen werden von den Heartbeat-Nachrichten von den überwachten Prozessorkernen zu einer Überwachungseinheit (FDU) übertragen. Dazu wurden zunächst die notwendigen Anpassungen an der Prozessorarchitektur beschrieben, die im Zuge dieser Dissertation notwendig waren. Ferner wurden alternative Lösungen diskutiert und die schlussendliche Realisierung der Anpassungen motiviert. Teile dieser Anpassungen waren zum einen das adaptive, TDMA-basierende Sendemuster, um Heartbeat-Nachrichten untereinander zu Isolieren. Zum anderen wurde das Netz in zwei virtuelle Kanäle unterteilt und diese unterschiedlich priorisiert. Heartbeat-Nachrichten mit hoher Priorität werden mit dieser QoS-Implementierung allen Anwendungsnachrichten beim Verarbeiten im Router vorgezogen. Beide Anpassungen ergaben zusammen, die geforderte Isolation der Heartbeat-Nachrichten.

Die aus der Priorisierung entstandenen Einflüsse der Heartbeat-Nachrichten auf Anwendungsnachrichten wurde anschließend untersucht und Engpässe identifiziert, welche bei der Verwendung der XY-Strategie in FDU-Nähe entstanden. Um potentielle Engpässe dieser Heartbeat-Nachrichten im Kommunikationsnetz zu vermindern, wurde in dieser Dissertation zusätzlich eine neue dimensions-orientierte Routing-Strategie entwickelt; die Staircase-Strategie. Unter Verwendung dieser Strategie verteilt sich die Last der Heartbeat-Nachrichten besser auf die vorhandenen Ressourcen des prozessorinternen Kommunikationsnetzes.

Mit Hilfe der Sendemuster *Random*, *Transpose*, *Bit-Reversal* und *Hot-Spot* für Anwendungsnachrichten wurde evaluiert, dass Heartbeat-Nachrichten im Durchschnitt keinen signifikanten Einfluss auf Anwendungsnachrichten in Bezug auf Durchsatz und Latenz haben. Mit Ausnahme der Verwendung des Sendemusters Hot-Spot, zeigt das Netz darüber hinaus auch nach dem Sättigungspunkt eine stabile Leistung, ohne Einbußen beim Durchsatz. Auch die durchschnittliche Latenz der Anwendungsnachrichten wies in Gegenwart der Heartbeat-Nachrichten keine signifikanten Unterschiede zu den Versuchen ohne Heartbeat-Nachrichten auf.

Einen deutlichen Vorteil ergab jedoch die Untersuchung der maximalen Verzögerung der Anwendungsnachrichten. Unter Verwendung der Staircase-Strategie konnte für die Sendemuster Random und Transpose die maximale Verzögerung gesenkt werden. Im Vergleich mit der XY-Strategie sanken die Wartezeiten um etwa 25 – 30%. Für das Sen-

demuster Hot-Spot hingegen, wurden bei beiden Routing-Strategien nahezu identische maximale Wartezeiten gemessen. Sowohl für die XY-, als auch die Staircase-Strategie liegen die gemessenen Verzögerung um etwa 20% über dem Referenzwert ohne Heartbeat-Nachrichten.

Das Bit-Reversal stellte sich, hinsichtlich der Staircase-Strategie, als problematisch heraus. Verglichen mit der XY-Strategie, lagen hier die maximalen Wartezeiten durch die Staircase-Strategie etwa 30% höher. Dieses Resultat war vor allem darauf zurückzuführen, dass das Sendemuster Bit-Reversal einen Großteil der anfallenden Kommunikation vom Zentrum des Netzes entfernt hielt. Damit konnte die Staircase-Strategie keinen Nutzen aus der breiteren Nutzung der Netzressourcen schöpfen.

Zusammenfassend bewertet, konnte die These untermauert werden, dass die gleichmäßigere Verteilung der Heartbeat-Nachrichten im Netz einen positiven Einfluss auf die Anwendungsnachrichten in Bezug auf die maximalen Wartezeiten haben.

Der zweite Teil dieser Arbeit präsentiert ein neues, leichtgewichtiges Verfahren, bei dem aus dem zeitlichem Verhalten der Heartbeat-Nachrichten, unter Berücksichtigung der verwendeten Routing-Strategie, Fehler im Kommunikationsnetz lokalisiert werden können. Durch den priorisierten Versand der Heartbeat-Nachrichten und den Erweiterungen des Heartbeat-Mechanismus, war es möglich, anhand der verspäteten Anlieferung einer Heartbeat-Nachricht und deren Route durch das Kommunikationsnetz, den Fehler im ersten Schritt einzugrenzen. Dabei wird die Route der Heartbeat-Nachricht durch eine deterministische Routing-Strategie (entweder XY- oder Staircase-Routing) vorgegeben. Durch den Erhalt zusätzlicher verspäteter Heartbeat-Nachrichten kann der Fehler weiter eingegrenzt werden. Dieses Eingrenzen war mächtig genug, um einzelne Fehler exakt lokalisieren zu können.

Hierfür musste die Basisarchitektur des TERAFLUX-Prozessors nicht signifikant erweitert werden. Die Grundvoraussetzung für die Lokalisierung wurde bereits in Form der Anpassungen für Heartbeat-Nachrichten auf dem Prozessor bereitgestellt. Jedoch waren Anpassungen des originalen Heartbeat-Mechanismus notwendig, um tote Winkel bei der Überwachung zu kompensieren.

Die toten Winkel entstanden durch Verbindungsleitungen, die von den Heartbeat-Nachrichten ungenutzt blieben und somit nicht nativ überwacht wurden. Durch das Verwenden des Poll-Verfahrens der FDU, das Überlappen der Cluster-Grenzen und den Einsatz alternierender Routing-Strategien konnten jedoch fast alle toten Winkel beseitigt werden. Eine besondere Form eines toten Winkels an den Prozessor-Ecken, machte eine Änderung der Hardware-Architektur in den betroffenen Eck-Routern notwendig. Hier gab es ohne die Erweiterung der Router-Pipeline keine Möglichkeit eine künstliche Verzögerung der Heartbeat-Nachrichten im Fehlerfall zu erzeugen. Erst mit dem Einsatz einer weiteren Pipeline-Stufe, die als zusätzlicher Puffer dient und damit die notwendige Verzögerung der Heartbeat-Nachricht erzeugt, konnten die toten Winkel aufgelöst werden.

Bei mehr als einem Fehler in einem logischen Cluster, ist die Lage der beiden Fehler entscheidend, um beide lokalisieren zu können. Eine Untersuchung dieser Fehlermuster führte jedoch zu zwei Erkenntnissen. Zum einen konnten aus den problematischen Fehlerkombinationen drei Gruppen, anhand ihrer Wirkung auf das Lokalisierungsverfahren, abgeleitet werden. Zusätzlich konnten die Gruppen in kritisch oder unkritisch unterschieden werden. Kritisch sind die Fehler, die in der Lage sind, andere Fehler zu maskieren und damit verhindern, dass die maskierten Fehler lokalisiert werden können. Unkritisch hingegen sind Fehlerkombinationen, bei denen ein weiterer Fehler erkannt wurde, der aber in Wirklichkeit nicht im Netz existiert (*Phantom-Fehler*).

Zum anderen hat die Untersuchung gezeigt, dass das zeitliche Auftreten der Fehler einen wesentlichen Einfluss auf die Lokalisierung hat. Treten die Fehler simultan im Netz auf, sind $8-10\%$ der Fehlerkombinationen problematisch (abhängig von der Cluster-Größe), wobei mindestens einer der entstandenen Fehler nicht lokalisiert werden kann. Zusätzlich nimmt aber der prozentuale Anteil der problematischen Fehlerkombinationen mit der Größe des Clusters ab. Besonders deutlich wird dies bei den sukzessiv auftretenden Fehlern. Neben dem deutlich geringeren Anteil problematischer Fehlermuster von maximal $2,3\%$ (Cluster-Größe: 5×5), sinkt der Anteil auf $1,6\%$ (Cluster-Größe: 7×7) mit größer werdenden Clustergrößen.

Abschließend wird für eine mögliche toroidale Netztopologie eine Diskussion geführt, welche als Alternative zum 2D-Gitter des TERAFLUX-Prozessors in Betracht gezogen wurde. Die weiteren Verbindungleitungen (Wrap-Around Link, WAL) haben sich auf mehreren Ebenen als problematisch dargestellt. Die bisher verwendeten Routing-Strategien beziehen die WALs nicht mit in die Überwachung der FDUs ein, sodass ein neuer toter Winkel an den Randbereichen des Prozessors entsteht. Zusätzlich kommt hinzu, dass die WALs nun neue Abhängigkeitsverhältnisse im Netz schaffen. Die zyklischen Abhängigkeiten des 2D-Gitters sind auf Richtungsänderungen in zwei Dimensionen bezogen. In toroidalen Netzen jedoch, existieren bereits in einer einzelnen Dimension solche zyklischen Abhängigkeiten. Damit mussten weitere Anpassungen an den Routern selbst vorgenommen werden, um Deadlocks und Livelocks im fehlerfreien Fall zu verhindern.

5.1. Ausblick

Durch die Untersuchung multipler Fehler und im Hinblick auf das verwendete Turn-Model, um Fehler im Netz zu tolerieren, kann das hier vorgestellte Verfahren weiterentwickelt werden, um robuster gegen multiple Fehler zu werden.

Die in den Grundlagen beschriebenen Probleme bei Deadlock-freien Routing-Strategien können auf verschiedene Arten gelöst werden. Ein vielversprechender Ansatz dazu ist das *Q Routing* [Boyan und Littman 1994], basierend auf dem Q-Learning-Konzept [Watkins und Dayan 1992] des maschinellen Lernens. Die Routing-Strategie wurde entwickelt, um

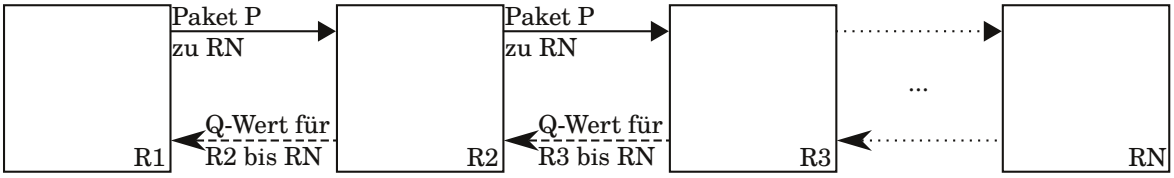


Abb. 5.1.: Pakettransfer und Quittierung durch Q-Werten

globales Wissen über die Topologie eines Netzes zu erzeugen und ähnelt dem *Distance Vector Routing*, bei dem die Routing-Strategie alle Pakete auf dem kürzesten Weg zum Ziel überträgt.

Das Q Routing verwendet dazu, in den aktuellsten Weiterentwicklungen [Feng 2010; Radetzki 2011], eine Routing-Tabelle auf jedem Router. Eine solche Routing-Tabelle beinhaltet für jeden Prozessorkern einen Eintrag, mit jeweils einem *Q-Wert* für jede Ausgangsleitung. Die Q-Werte geben dabei die zu erwartende Güte an, die eine Verbindungsleitung bereitstellt, um das aktuelle Paket an den Empfänger weiterzuleiten. Abgesetzte Pakete werden, wie in Abbildung 5.1 illustriert, von dem Empfänger-Router mit einem neuen Q-Wert quittiert (ähnlich wie bei einem ACK/NACK-Verfahren zur Fehlererkennung). Als Quittung verwendet der Empfänger-Router den Q-Wert seiner lokalen, favorisierten Ausgangsleitung, um das Paket weiter zu übertragen.

Der ursprüngliche Router nimmt so jede erhaltende Quittierung auf und aktualisiert seine lokalen Q-Werte schrittweise für die entsprechenden Ausgangsleitungen. Für die Aktualisierung wird die *Q-Funktion* [Watkins und Dayan 1992] verwendet, die den alten und den neuen Q-Wert kombiniert. Da alle Router dieser Art des Informationsaustausches folgen, werden Defekte im Netz, schrittweise an alle Router propagiert. Da die Aktualisierungen iterativ mit jedem Paket durchgeführt werden, konvergieren die Q-Werte in der Routing-Tabelle. Dies impliziert jedoch auch, dass transiente und intermittierende Fehler nicht mit diesem Verfahren behandelt werden können. Wobei letztere auf die Konvergenz der Q-Werte störend wirken.

Bei Feng *et al.* entspricht die Güte der Anzahl an Hops, die ein Paket benötigt, wenn es über eine bestimmte Ausgangsleitung übertragen wird. Für einen Prozessor mit 1024 Kernen (32×32) würde dies etwa 3-4MiB Tabellenspeicher auf dem Chip erfordern[1]. Dies klingt zunächst nach einer signifikanten Belastung der verfügbaren Chip-Fläche. Betrachtet man jedoch Intel's Projektionen für zukünftige Prozessoren [Borkar und Chien 2011], verfügen diese Prozessoren über etwa 80MiB Cache-Speicher[2]. Nach Radetzki wird zusätzliches Grundwissen über die Topologie vorausgesetzt, sodass in den Routing-Tabellen nur unterschieden werden muss, ob eine Ausgangsleitung die Distanz zum Ziel

[1] Abhängig von der Netzgröße, Topologie und der maximal erlaubten Pfädlänge (hier 255 Hops).
[2] Intel rechnet dort mit der Faustregel, dass aus Effizienzgründen etwa 50% des Chips für Cache-Speicher zur Verfügung steht.

verkürzt oder nicht. Die damit verwendeten booleschen Werte ließen den Speicherverbrauch auf 512KiB sinken.

Diese Routing-Strategien zeigen sehr gute Anpassungsfähigkeiten, wenn im Netz Verbindungsleitungen oder Router ausfallen [Feng 2010; Radetzki 2011]. Die Anpassungsfähigkeit und die Routing-Tabellen sind Eigenschaften, die das Q-Routing für eine Fehlerlokalisierung im Netz attraktiv machen. Durch die Q-Werte befinden sich in allen Tabellen abstrakte Informationen über den gesamten Netzzustand.

Um einer FDU-ähnlichen Einheit Zugriff auf diese Tabelle zu ermöglichen, wären drei Methoden denkbar:

(i) Ein spezielles Datenpaket kann den Router veranlassen, seine aktuelle Tabelle der angeschlossenen FDU per Nachricht zu übermitteln.

(ii) Jeder Prozessorkern verfügt über eine separate Datenleitung zum Router, um die Tabelle vom Router herunterladen zu können.

(iii) Der Prozessorkern und der Router teilen sich den Speicherbereich der Tabellen, sodass die FDU in der Lage ist, direkt die Daten der Tabelle zu laden.

Eine einzelne Tabelle allein wird jedoch nicht ausreichen, um den Netzzustand aus den vorliegenden Q-Werten vollständig zu rekonstruieren. Daher wird vorgeschlagen, die Tabellen mit den einzelnen FDUs zu teilen. Denn jede FDU hat, durch die lokalen Q-Werte, jeweils eine ganz eigene Perspektive auf das Netz. Kombiniert man das lokale Wissen der FDUs, könnten ähnliche Ansätze entwickelt werden, wie sie hier in der Dissertationsarbeit vorgestellt wurden.

Die Erwartung an dieses Verfahren läge vor allem darin, sich von den Heartbeat-Nachrichten zu entkoppeln, was ebenfalls die Notwendigkeit der Isolation der einzelnen Nachrichten-Typen aufhebt und die FDU hinsichtlich der Prozessorkern-Fehlertoleranz flexibler bei der Implementierung anderer Verfahren machen würde.

Literaturverzeichnis

[Abdel-Khalek 2011] R. Abdel-Khalek. „Functional correctness for CMP interconnects“. In: *Computer Design (ICCD), 2011 IEEE 29th International Conference on*. 2011, Seiten 352–359. DOI: 10.1109/ICCD.2011.6081423.

[Adriahantenaina 2003] A. Adriahantenaina. „SPIN: a scalable, packet switched, on-chip micro-network“. In: *Design, Automation and Test in Europe Conference and Exhibition, 2003*. 2003, 70–73 suppl. DOI: 10.1109/DATE.2003.1253808.

[Bell 2008] S. Bell. „TILE64 - Processor: A 64-Core SoC with Mesh Interconnect“. In: *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*. 2008, Seiten 88–598. DOI: 10.1109/ISSCC.2008.4523070.

[Benini 2002] L. Benini und G. De Micheli. „Networks on Chips: A New SoC Paradigm“. In: *Computer* 35.1 (Jan. 2002), Seiten 70–78. ISSN: 0018-9162. DOI: 10.1109/2.976921. URL: http://dx.doi.org/10.1109/2.976921.

[Borkar 2005] S. Borkar. „Designing Reliable Systems from Unreliable Components: the Challenges of Transistor Variability and Degradation“. In: *IEEE Micro* 25.6 (2005). ISSN: 0272-1732. DOI: 10.1109/MM.2005.110.

[Borkar 2007] S. Borkar. „Thousand Core Chips: A Technology Perspective“. In: *Proceedings of the 44th Annual Design Automation Conference*. DAC '07. New York, NY, USA: ACM, 2007, Seiten 746–749. ISBN: 978-1-59593-627-1. DOI: 10.1145/1278480.1278667. URL: http://doi.acm.org/10.1145/1278480.1278667.

[Borkar 2011] S. Borkar und A. Chien. „The Future of Microprocessors“. In: *Commun. ACM* 54.5 (Mai 2011), Seiten 67–77. ISSN: 0001-0782. DOI: 10.1145/1941487.1941507. URL: http://doi.acm.org/10.1145/1941487.1941507.

[Boyan 1994] J. Boyan und M. Littman. „Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach“. In: *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1994, Seiten 671–678.

[Chen 2012] Xi Chen. „In-network Monitoring and Control Policy for DVFS of CMP Networks-on-Chip and Last Level Caches“. In: *Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*. Mai 2012, Seiten 43–50. DOI: 10.1109/NOCS.2012.12.

[Constantinescu 2003] C. Constantinescu. „Trends and challenges in VLSI circuit reliability“. In: *Micro, IEEE* 23.4 (2003), Seiten 14–19. ISSN: 0272-1732. DOI: 10.1109/MM.2003.1225959.

[Cota 2008] E. Cota. „A High-Fault-Coverage Approach for the Test of Data, Control and Handshake Interconnects in Mesh Networks-on-Chip“. In: *Computers, IEEE Transactions on* 57.9 (2008), Seiten 1202–1215. ISSN: 0018-9340. DOI: 10.1109/TC.2008.62.

[Dall'Osso 2003] M. Dall'Osso. „Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs“. In: *Computer Design, 2003. Proceedings. 21st International Conference on*. 2003, Seiten 536–539. DOI: 10.1109/ICCD.2003.1240952.

[Dally 1992] W.J. Dally. „Virtual-channel flow control“. In: *Parallel and Distributed Systems, IEEE Transactions on* 3.2 (1992), Seiten 194–205. ISSN: 1045-9219. DOI: 10.1109/71.127260.

[Dally 2001] W.J. Dally und B. Towles. „Route packets, not wires: on-chip interconnection networks“. In: *Design Automation Conference, 2001. Proceedings*. 2001, Seiten 684–689. DOI: 10.1109/DAC.2001.156225.

[Dally 2003] William Dally und Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003. ISBN: 0122007514.

[David 2011] R. David. „Dynamic power management of voltage-frequency island partitioned Networks-on-Chip using Intel's Single-chip Cloud Computer“. In: *Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on*. 2011, Seiten 257–258.

[Dielissen 2003] John Dielissen. „Concepts and Implementation of the Philips Network-on-Chip“. In: *IN IP-BASED SOC DESIGN*. 2003.

[Dinechin 2013a] B.D. de Dinechin. „A clustered manycore processor architecture for embedded and accelerated applications“. In: *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. 2013, Seiten 1–6. DOI: 10.1109/HPEC.2013.6670342.

[Dinechin 2013b] Benoît Dupont de Dinechin. „A Distributed Run-Time Environment for the Kalray MPPA-256 Integrated Manycore Processor“. In: *Procedia Computer Science* 18.0 (2013). 2013 International Conference on Computational Science, Seiten 1654 –1663. ISSN: 1877-0509. DOI: http://dx.doi.org/10.1016/j.procs.2013.05.333. URL: http://www.sciencedirect.com/science/article/pii/S1877050913004766.

[Duato 1993] J. Duato. „A new theory of deadlock-free adaptive routing in wormhole networks“. In: *Parallel and Distributed Systems, IEEE Transactions on* 4.12 (1993), Seiten 1320–1331. ISSN: 1045-9219. DOI: 10.1109/71.250114.

[Duato 1995] J. Duato. „A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks“. In: *Parallel and Distributed Systems, IEEE Transactions on* 6.10 (1995), Seiten 1055–1067. ISSN: 1045-9219. DOI: 10.1109/71.473515.

[Duato 1996] J. Duato. „A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks“. In: *Parallel and Distributed Systems, IEEE Transactions on* 7.8 (1996), Seiten 841–854. ISSN: 1045-9219. DOI: 10.1109/71.532115.

[Duato 2002] Jose Duato. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., 2002. ISBN: 1558608524.

[Dziurzanski 2013] P. Dziurzanski und T. Maka. „Core Mapping into an Irregular Network on Chip - Features Extraction System for Automatic Speech Recognition Case Study“. In: *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. 2013, Seiten 494–498. DOI: 10.1109/PDP.2013.79.

[Feng 2010] Chaochao Feng. „A Reconfigurable Fault-tolerant Deflection Routing Algorithm Based on Reinforcement Learning for Network-on-chip“. In: *Proceedings of the Third International Workshop on Network on Chip Architectures*. NoCArc '10. Atlanta, Georgia: ACM, 2010, Seiten 11–16. ISBN: 978-1-4503-0397-2. DOI: 10.1145/1921249.1921254. URL: http://doi.acm.org/10.1145/1921249.1921254.

[Fick 2009] D. Fick. „Vicis: A reliable network for unreliable silicon“. In: *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*. 2009, Seiten 812–817.

[Flich 2010] José Flich und Davide Bertozzi. *Design Network On-Chip Architecures in the Nanoscale Era*. Herausgegeben von José Flich und Davide Bertozzi. Chapman & Hall/CRC Computational Science, 2010.

[Furber 2008] S. Furber. „The Future of Computer Technology and Its Implications for the Computer Industry“. In: *Comput. J.* 51.6 (Nov. 2008), Seiten 735–740. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxn022. URL: http://dx.doi.org/10.1093/comjnl/bxn022.

[Garbade 2013] A. Garbade. „Fault Localization in NoCs Exploiting Periodic Heartbeat Messages in a Many-Core Environment“. In: *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*. 2013, Seiten 791–795. DOI: 10.1109/IPDPSW.2013.150.

[Gebhardt 2011] Daniel Gebhardt. „Link pipelining strategies for an application-specific asynchronous NoC“. In: *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*. NOCS '11. Pittsburgh, Pennsylvania: ACM, 2011, Seiten 185–192. ISBN: 978-1-4503-0720-8. DOI: 10.1145/1999946.1999976. URL: http://doi.acm.org/10.1145/1999946.1999976.

[Ghiribaldi 2011] A. Ghiribaldi. „System-level infrastructure for boot-time testing and configuration of networks-on-chip with programmable routing logic“. In: *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*. 2011, Seiten 308 –313. DOI: 10.1109/VLSISoC.2011.6081597.

[Ghiribaldi 2013] Alberto Ghiribaldi. „A Complete Self-testing and Self-configuring NoC Infrastructure for Cost-effective MPSoCs“. In: *ACM Trans. Embed. Comput. Syst.* 12.4 (Juli 2013), 106:1–106:29. ISSN: 1539-9087. DOI: 10.1145/2485984.2485994. URL: http://doi.acm.org/10.1145/2485984.2485994.

[Ghofrani 2012] A.-A. Ghofrani. „Comprehensive online defect diagnosis in on-chip networks“. In: *VLSI Test Symposium (VTS), 2012 IEEE 30th*. 2012, Seiten 44–49. DOI: 10.1109/VTS.2012.6231078.

[Glass 1992] C.J. Glass und L.M. Ni. „The Turn Model for Adaptive Routing“. In: *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*. 1992, Seiten 278–287. DOI: 10.1109/ISCA.1992.753324.

[Goossens 2003] Kees Goossens. „Networks on chip“. In: Herausgegeben von Axel Jantsch und Hannu Tenhunen. Hingham, MA, USA: Kluwer Academic Publishers, 2003. Kapitel Guaranteeing the quality of services in networks on chip, Seiten 61–82. ISBN: 1-4020-7392-5. URL: http://dl.acm.org/citation.cfm?id=903951.903956.

[Goossens 2005] Kees Goossens. „Æthereal Network on Chip:Concepts, Architectures, and Implementations“. In: *IEEE Design & Test of Computers* 22 (2005), Seiten 414–421. ISSN: 0740-7475. DOI: http://doi.ieeecomputersociety.org/10.1109/MDT.2005.99.

[Grecu 2006a] C. Grecu. „BIST for network-on-chip interconnect infrastructures“. In: *VLSI Test Symposium, 2006. Proceedings. 24th IEEE*. 2006, 6 pp.–35. DOI: 10.1109/VTS.2006.22.

[Grecu 2006b] C. Grecu. „On-line fault detection and location for NoC interconnects“. In: *On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International*. 2006. DOI: 10.1109/IOLTS.2006.44.

[Howard 2010] J. Howard. „A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS“. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*. 2010, Seiten 108–109. DOI: 10.1109/ISSCC.2010.5434077.

[Hu 2004] Jingcao Hu und R. Marculescu. „DyAD - smart routing for networks-on-chip“. In: *Design Automation Conference, 2004. Proceedings. 41st*. 2004, Seiten 260–263.

[ITRS 2007] ITRS. *International Technology Roadmap for Semiconductors 2007 Edition*. Website. Available online at http://www.itrs.net, visited on August 12th 2011. 2007.

[ITRS 2011] ITRS. *International Technology Roadmap for Semiconductors 2011 Edition*. Website. Available online at http://www.itrs.net, visited on September 20th 2013. 2011.

[Kahng 2009] A.B. Kahng. „ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration“. In: *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.* 2009, Seiten 423–428. DOI: 10.1109/DATE.2009.5090700.

[Kavaldjiev 2004] N. Kavaldjiev. „A virtual channel router for on-chip networks“. In: *SOC Conference, 2004. Proceedings. IEEE International*. 2004, Seiten 289–293. DOI: 10.1109/SOCC.2004.1362438.

[Kephart 2003] J.O. Kephart und D.M. Chess. „The Vision of Autonomic Computing“. In: *Computer* 36.1 (Jan. 2003), Seiten 41–50. ISSN: 0018-9162. DOI: 10.1109/MC.2003.1160055. URL: http://portal.acm.org/citation.cfm?id=642200.

[Kim 2004] Daewook Kim. „CDMA-based network-on-chip architecture“. In: *Circuits and Systems, 2004. Proceedings. The 2004 IEEE Asia-Pacific Conference on*. Band 1. 2004, 137–140 vol.1. DOI: 10.1109/APCCAS.2004.1412711.

[Kim 1992] J.H. Kim und A. Chien. „An evaluation of planar-adaptive routing (PAR)“. In: *Parallel and Distributed Processing, 1992. Proceedings of the Fourth IEEE Symposium on*. 1992, Seiten 470–478. DOI: 10.1109/SPDP.1992.242708.

[Ludovici 2009] Daniele Ludovici. „Capturing Topology-Level Implications of Link Synthesis Techniques for Nanoscale Networks-on-Chip“. In: *Proceedings of the 19th ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI*. 2009, Seiten 125–128.

[Matos 2011] D. Matos. „Reconfigurable Routers for Low Power and High Performance“. In: *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 19.11 (2011), Seiten 2045–2057. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2010.2068064.

[Moore 1965] Gordon E. Moore. „Cramming more components onto integrated circuits“. In: *Electronics* 38.8 (1965).

[Murali 2005] S. Murali. „Analysis of error recovery schemes for networks on chips“. In: *Design Test of Computers, IEEE* 22.5 (2005), Seiten 434–442. ISSN: 0740-7475. DOI: 10.1109/MDT.2005.104.

[Ni 1993] L.M. Ni und P.K. McKinley. „A survey of wormhole routing techniques in direct networks“. In: *Computer* 26.2 (1993), Seiten 62–76. ISSN: 0018-9162. DOI: 10.1109/2.191995.

[Nicolaidis 1998] M. Nicolaidis und Y. Zorian. „On-Line Testing for VLSI—A Compendium of Approaches“. English. In: *Journal of Electronic Testing* 12.1-2 (1998), Seiten 7–20. ISSN: 0923-8174. DOI: 10.1023/A:1008244815697. URL: http://dx.doi.org/10.1023/A%3A1008244815697.

[Nightingale 2011] Edmund B. Nightingale. „Cycles, cells and platters: an empirical analysisof hardware failures on a million consumer PCs“. In: *Proceedings of the sixth conference on Computer systems*. EuroSys '11. Salzburg, Austria: ACM, 2011, Seiten 343–356. ISBN: 978-1-4503-0634-8. DOI: 10.1145/1966445.1966477. URL: http://doi.acm.org/10.1145/1966445.1966477.

[Palesi 2007] Maurizio Palesi. *Noxim - An Open Network-on-Chip Simulator*. 2007.

[Pande 2005a] P.P. Pande. „Design, synthesis, and test of networks on chips“. In: *Design Test of Computers, IEEE* 22.5 (2005), Seiten 404–413. ISSN: 0740-7475. DOI: 10.1109/MDT.2005.108.

[Pande 2005b] P.P. Pande. „Performance evaluation and design trade-offs for network-on-chip interconnect architectures“. In: *Computers, IEEE Transactions on* 54.8 (2005), Seiten 1025–1040. ISSN: 0018-9340. DOI: 10.1109/TC.2005.134.

[Park 2006] Dongkook Park. „Exploring Fault-Tolerant Network-on-Chip Architectures“. In: *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*. 2006, Seiten 93–104. DOI: 10.1109/DSN.2006.35.

[Pekkarinen 2011] E. Pekkarinen. „A set of traffic models for Network-on-Chip benchmarking“. In: *System on Chip (SoC), 2011 International Symposium on*. 2011, Seiten 78–81. DOI: 10.1109/ISSOC.2011.6089221.

[Prodromou 2012] A. Prodromou. „NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures“. In: *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*. 2012, Seiten 60–71. DOI: 10.1109/MICRO.2012.15.

[Radetzki 2011] M. Radetzki. „Fault-Tolerant Differential Q Routing in Arbitrary NoC Topologies“. In: *Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on*. 2011, Seiten 33–40. DOI: 10.1109/EUC.2011.36.

[Salihundam 2010] P. Salihundam. „A 2Tb/s 6×4 mesh network with DVFS and 2.3Tb/s/W router in 45nm CMOS“. In: *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*. 2010, Seiten 79–80. DOI: 10.1109/VLSIC.2010.5560277.

[Salminen 2010] Erno Salminen. *OCP-IP Network-on-chip benchmarking workgroup*. White Paper. 2010.

[Schley 2013] G. Schley. „Fault Localizing End-to-End Flow Control Protocol for Networks-on-Chip“. In: *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. 2013, Seiten 454–461. DOI: 10.1109/PDP.2013.74.

[Schlingmann 2011] Sebastian Schlingmann. „Connectivity-Sensitive Algorithm for Task Placement on a Many-Core Considering Faulty Regions“. In: *Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*. PDP '11. Washington, DC, USA: IEEE Computer Society, 2011, Seiten 417–422. ISBN: 978-0-7695-4328-4. DOI: 10.1109/PDP.2011.58. URL: http://dx.doi.org/10.1109/PDP.2011.58.

[Schroeder 2010] B. Schroeder und G.A. Gibson. „A Large-Scale Study of Failures in High-Performance Computing Systems“. In: *Dependable and Secure Computing, IEEE Transactions on* 7.4 (2010), Seiten 337–350. ISSN: 1545-5971. DOI: 10.1109/TDSC.2009.4.

[Schroeder 2009] B. Schroeder. „DRAM errors in the wild: a large-scale field study“. In: *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. SIGMETRICS '09. Seattle, WA, USA: ACM, 2009, Seiten 193–204. ISBN: 978-1-60558-511-6. DOI: 10.1145/1555349.1555372. URL: http://doi.acm.org/10.1145/1555349.1555372.

[Shamshiri 2011] S. Shamshiri. „End-to-end error correction and online diagnosis for on-chip networks“. In: *Test Conference (ITC), 2011 IEEE International*. 2011, Seiten 1–10. DOI: 10.1109/TEST.2011.6139156.

[Sun 2012] Chen Sun. „DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling“. In: *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*. 2012, Seiten 201–210. DOI: 10.1109/NOCS.2012.31.

[Wang 2001] P. C Wang und R.G. Filippi. „Electromigration threshold in copper interconnects“. In: *Applied Physics Letters* 78.23 (2001), Seiten 3598–3600. ISSN: 0003-6951. DOI: 10.1063/1.1371251.

[Watkins 1992] ChristopherJ.C.H. Watkins und Peter Dayan. „Q-learning“. English. In: *Machine Learning* 8.3-4 (1992), Seiten 279–292. ISSN: 0885-6125. DOI: 10.1007/BF00992698. URL: http://dx.doi.org/10.1007/BF00992698.

[Weis 2011] Sebastian Weis. „Towards Fault Detection Units as an Autonomous Fault Detection Approach for Future Many-Cores“. In: *ARCS Workshops 2011 Proceedings (SCAFT)*. 2011. ISBN: 978-3-8007-3333-0.

[Wentzlaff 2007] D. Wentzlaff. „On-Chip Interconnection Architecture of the Tile Processor“. In: *Micro, IEEE* 27.5 (2007), Seiten 15–31. ISSN: 0272-1732. DOI: 10.1109/MM.2007.4378780.

[Wijngaart 2011] Rob F. van der Wijngaart. „Light-weight Communications on Intel's Single-chip Cloud Computer Processor“. In: *SIGOPS Oper. Syst. Rev.* 45.1 (Feb. 2011), Seiten 73–83. ISSN: 0163-5980. DOI: 10.1145/1945023.1945033. URL: http://doi.acm.org/10.1145/1945023.1945033.

A. Beispiel Nachrichten Isolierung

A.1. Arbitrierung der Heartbeat-Nachrichten: Blockierend

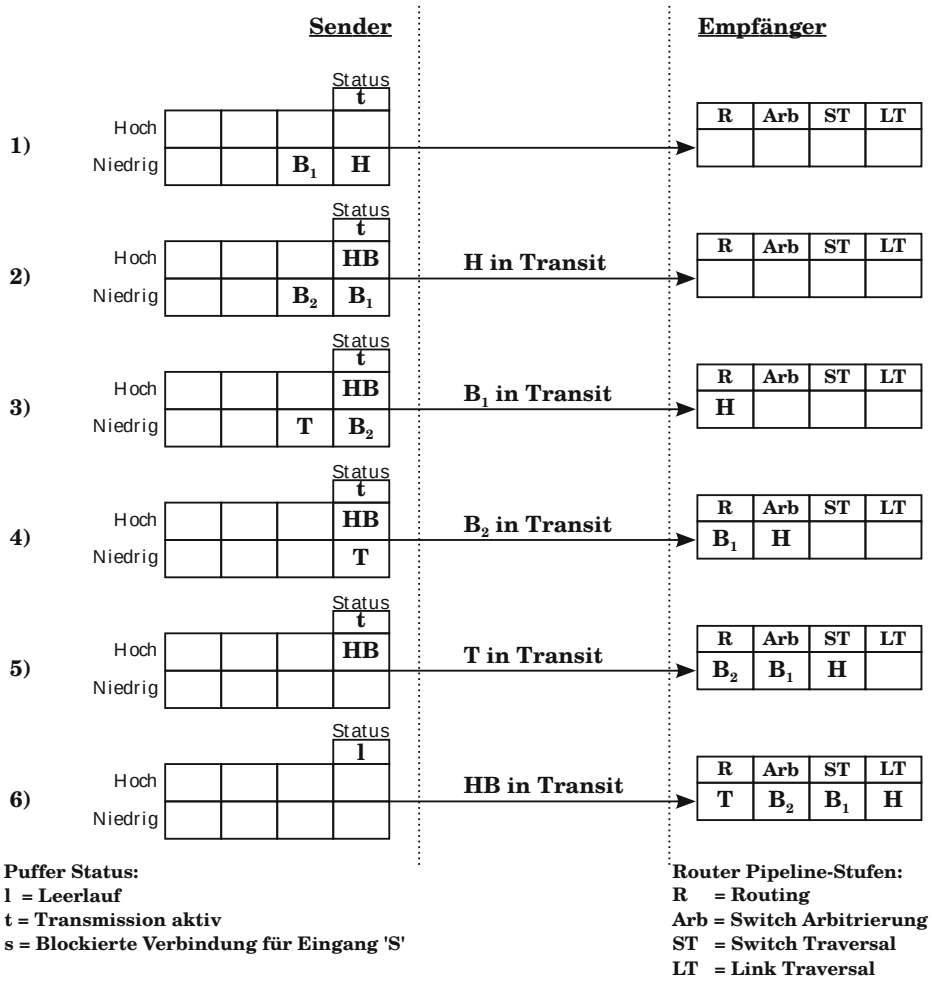


Abb. A.1.: Die Heartbeat-Nachricht wird blokiert, solang die Anwendungsnachricht zum Empfänger übermittelt wird. Das löst die Isolation zwischen Heartbeat- und Anwendungsnachrichten auf und der Determinismuss ist nicht länger gegeben.

A.2. Arbitrierung der Heartbeat-Nachrichten: nicht Blockierend

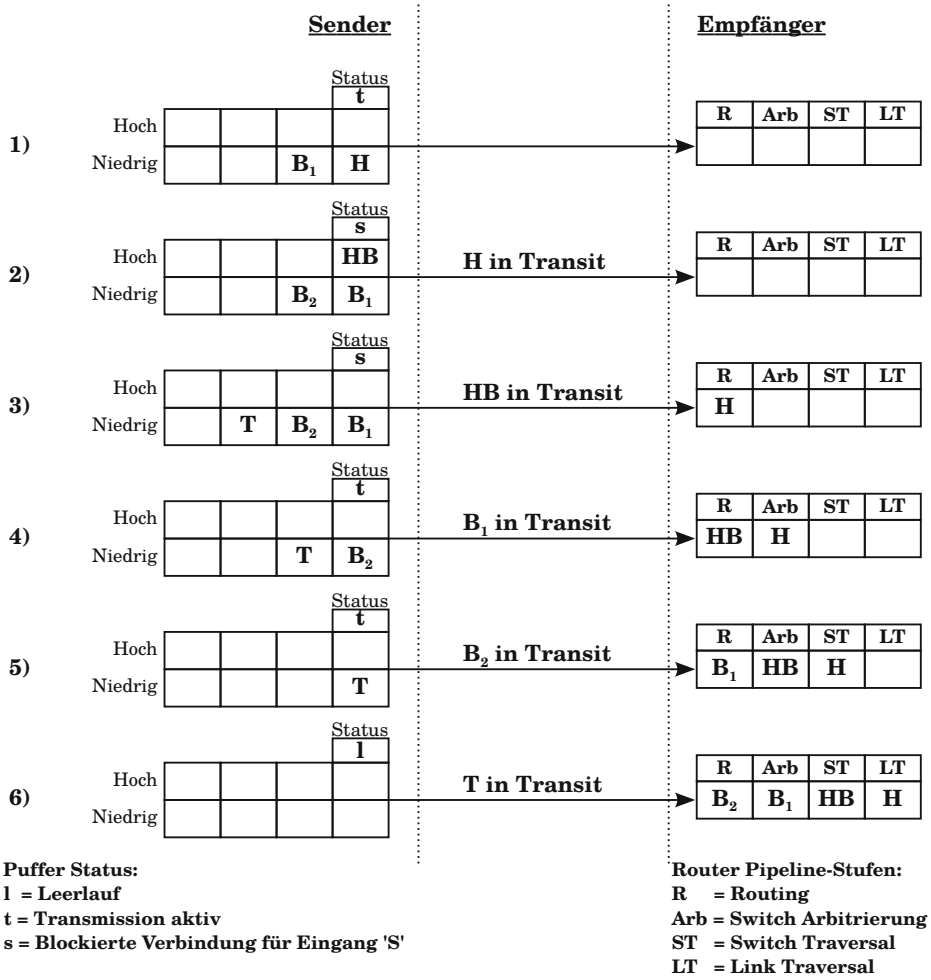


Abb. A.2.: Die Heartbeat-Nachricht werden zwischen den Anwendungsnachrichten zum Empfänger übertragen. Die Isolation belibt bestehen und bewart den Determinismus der Heartbeat-Nachrichten.

Danksagung

Diese wissenschaftliche Arbeit ist nicht das Werk eines einzelnen, viel mehr ist sie das Resultat aus vielen Arbeitsstunden, zahlreichen konstruktiven Diskussionen am Lehrstuhl und dem Rückhalt von Familie und Freunden.

Herrn Prof. Dr. Theo Ungerer möchte ich für die intensive, aufgeschlossene und professionelle Betreuung meiner Arbeit an der Universität Augsburg herzlich danken. Darüber hinaus hat er sich als erster Gutachter meiner Arbeit bereitgestellt, worüber ich ebenfalls sehr dankbar bin. Auch meinem Zweitgutachter Herrn Prof. Dr. Rudi Knorr möchte ich für seine fruchtbaren und äußerst hilfreichen Diskussionsbeiträge danken.

Einen ganz besonderen Dank richte ich an meine lieben Kollegeninnen und Kollegen des Lehrstuhls. Ohne die außerordentlich gute Zusammenarbeit und die Bereitstellung des breiten Knowhows, wäre diese Arbeit in der Form nicht möglich gewesen. Hervorheben möchte ich daher besonders meinen Kollegen und Mitdoktorand Herrn Sebastian Weis. Mit ihm zusammen im Projekt zu arbeiten sowie die zahlreichen wertvollen Diskussionen, waren mir ein großes Vergnügen und halfen die Qualität dieser Arbeit zu steigern.

Doch nichts war so ermutigend und motivierend wie der Rückhalt aus meiner Familie und meinen Freunden. Vor allen anderen gilt mein Dank Katja Kaiser. Sie hat mir, trotz der vielen Jahre und der großen räumlichen Distanz, zu jeder Zeit kompromisslos beigestanden. Ihre unermüdliche Kraft, Geduld und Zuversicht hat mich immer wieder aufs Neue angetrieben mein Ziel weiter zu verfolgen und es zu erreichen. Mein Dank ihr gegenüber vermag ich nicht in Worte zu fassen.

Auch meinen Eltern Angelika und Martin, sowie meinen drei Brüdern Martin, Maik und Malte gebührt großer Dank. Ganz nach dem Motto “Gemeinsam sind wir stark” wurde mir ein riesiger Rückhalt zuteil und allesamt halfen mir warmherzig, mich weiter auf Kurs zu halten. Einen speziellen Dank richte ich auch an meinen lieben Freund und jahrelangen Weggefährten Matthias Würthele. Ohne ihn hätte ich wohl nie das Studium gewählt, dass mich schlussendlich zu dieser Arbeit geführt hat.